ManiTaskGen: A Comprehensive Task Generator for Benchmarking and Improving Vision-Language Agents on Embodied Decision-Making

Anonymous ICCV submission

Paper ID HRSIC 10

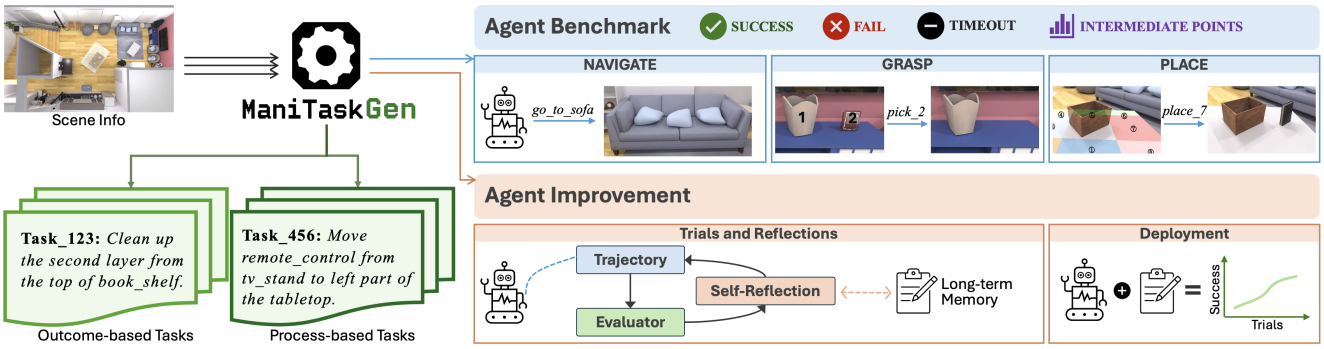

Figure 1. **Framework Overview.** ManiTaskGen is a universal system that generates a comprehensive set of feasible mobile manipulation tasks given arbitrary scene. These tasks facilitate automatic benchmarking and the improvement of embodied decision-making agents.

Abstract

Building embodied agents capable of accomplishing arbitrary tasks is a core objective towards achieving embodied artificial general intelligence (E-AGI). While recent work has advanced such general robot policies, their training and evaluation are often limited to tasks within specific scenes, involving restricted instructions and scenarios. Existing benchmarks also typically rely on manual annotation of limited tasks in a few scenes. We argue that exploring the full spectrum of feasible tasks within any given scene is crucial, as they provide both extensive benchmarks for evaluation and valuable resources for agent improvement. Towards this end, we introduce ManiTaskGen, a novel system that automatically generates comprehensive, diverse, feasible mobile manipulation tasks for any given scene. The generated tasks encompass both process-based, specific instructions (e.g., "move object from X to Y") and outcome-based, abstract instructions (e.g., "clear the table"). We apply ManiTaskGen to both simulated and real-world scenes, demonstrating the validity and diversity of the generated tasks. We then leverage these tasks to automatically construct benchmarks, thoroughly evaluating the embodied decision-making capabilities of agents built upon existing vision-language models (VLMs). Furthermore, we propose a simple yet effective method that utilizes ManiTaskGen tasks to enhance embodied decision-making. Overall, this work presents a universal task generation framework for arbitrary scenes, facilitating both benchmarking and improvement of embodied decision-making agents.

1. Introduction

Consider an embodied agent endowed with robust primitive skills for mobile manipulation: the ability to navigate to any accessible location, grasp any movable object, and place it wherever it fits. A fundamental question then arises: *what is the full extent of the task space this agent can successfully address in a given environment?* This space appears infinite. Despite this immense potential task space, recent efforts have focused on developing general-purpose embodied agents [2, 5, 7, 11, 16, 35, 36] capable of completing any feasible task within it. A promising direction leverages VLMs [4, 26, 38] for high-level decision-making, followed by either the explicit composition of skill primitives [13, 14, 17] or the implicit integration with underlying action modules to predict joint-level actions—the latter thread is commonly referred to as Vision-Language-Action (VLA) models [6, 7, 20]. To evaluate such agents, numer-

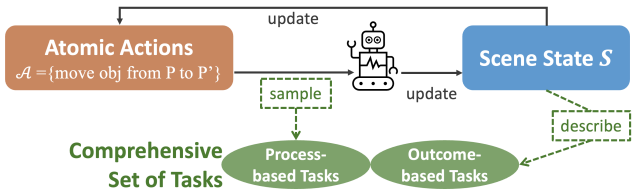

Figure 2. **Agent-Env Interaction Cycle.** This cycle provides a fundamental depiction of how an embodied agent executes a task within a scene. Accordingly, we categorize all tasks into two distinct types: process- and outcome-based tasks.

ous benchmarks [9, 23, 24, 44, 47] have also been proposed to assess their decision-making capabilities.

Despite these efforts, existing work faces significant limitations regarding the tasks used in both policy training and benchmark design. Common practice [8, 14, 22, 44] involves manually selecting a few scenes and authoring a limited number of scene-specific tasks related to objects within those scenes, which serve as training or testing objectives for embodied agents. Such manual approaches are labor-intensive and require considerable human effort to design suitable tasks, resulting in prohibitively high scaling costs and making it nearly impossible to generalize task creation to the diverse range of scenes encountered in both simulated and real-world environments. This fundamental limitation creates a substantial gap between current agents and the ultimate goal of E-AGI: while the aim is an agent capable of universal generalization across diverse scenes and tasks, existing works are confined to training and evaluating the agents within limited scenarios and task variations.

To address these limitations, we present ManiTaskGen, a universal mobile manipulation task generator for arbitrary scenes. Given scene information (e.g., object poses, object bounding boxes, object mesh models, etc.), it automatically generates a comprehensive and diverse collection of feasible mobile manipulation tasks that are logically near-exhaustive for that specific scene. We ensure logical comprehensiveness by grounding task generation in a systematic analysis of the fundamental agent-environment interaction cycle, as conceptually illustrated in Fig. 2.

This cycle, where executing atomic actions (e.g., single object relocation) updates the scene state and determines subsequent available actions, inherently defines the space of all possible tasks. Based on this inherent structure, we rigorously categorize all possible tasks into two principal types: **process-based tasks**, capturing action sequences or trajectories (e.g., "move object A from X to Y"), and **outcome-based tasks**, representing reachable target states (e.g., "make the table clean"). ManiTaskGen employs distinct strategies for generating each type. Process-based tasks are generated by explicitly sampling and composing atomic action sequences, which are derived from a novel Receptacle-Aware 3D Scene Graph encoding all objects and fine-grained potential placements within the scene. Outcome-based tasks are generated using a hybrid template-based approach combined with VLM voting mechanism to produce diverse descriptions of plausible target states.

We assess the validity and diversity of the generated tasks by applying ManiTaskGen to both simulated environments (e.g., ReplicaCAD [33], AI2THOR [21]) and real-world scenes (e.g., SUN-RGBD [32]). Furthermore, we propose a framework for automatically constructing benchmarks in simulators using the generated tasks to assess embodied agent decision-making capabilities, and conduct extensive evaluations of existing VLMs using this benchmark. Finally, we design an improvement method based on Inference-time Reinforcement Learning [30, 45] to leverage ManiTaskGen tasks for enhancing the decision-making abilities of existing VLM agents.

In summary, this work makes the following contributions: (1) We propose ManiTaskGen, a universal system for generating comprehensive and diverse mobile manipulation tasks for arbitrary scenes. (2) Leveraging the automatically constructed benchmarks based on the generated tasks, we conduct an extensive evaluation of the embodied decision-making capabilities of current VLMs. (3) We demonstrate the utility of the ManiTaskGen tasks for enhancing embodied decision-making in current VLMs through a proposed inference-time RL method.

2. Related Work

Task Generation for Embodied Agents. Recent efforts have explored task generation for digital agents [10, 15, 19, 29] and augmenting RL objectives [12, 39]. Among these, ALFRED [31] is the most relevant to our work. It combines a procedural task planner with human-annotated task directives. However, ALFRED focuses on expanding task trajectories rather than task definitions and lacks a comprehensive coverage of both process- and outcome-based mobile manipulation tasks. Moreover, it is limited to specific scenes. In contrast, ManiTaskGen aims to generate a diverse and comprehensive set of tasks for arbitrary scenes, emphasizing scalability and variability in task formulation.

Datasets and Benchmarks for Embodied Agents. Numerous datasets and benchmarks have been proposed for training [5, 21, 27, 28, 34] and evaluating [22, 27, 31] embodied agents, including those tailored to LLM/VLM-based decision-making agents [8, 23, 24, 43]. A common limitation of these works is their reliance on manually annotated tasks confined to a finite set of predefined scenes. In contrast, ManiTaskGen introduces a general framework for generating rich tasks across arbitrary scenes. Beyond serving as a static dataset, it also provides dynamic evaluation and optimization platform for embodied decision-making agents, making it a more versatile resource.

3. Comprehensive Task Generation

3.1. Premise and Formalization

We first formalize the problem for generating comprehensive mobile manipulation tasks solvable by an agent equipped with fundamental navigate, pick, and place skills.

We define the scene state $\mathcal{S}$ by the states of all objects $\mathcal{O} = \{o_1, \ldots, o_N\}$ within it. The state of an object o_i is $s_i = (p_i, c_i)$, including its pose p_i and its containment state c_i. The containment state specifies the surface it is currently located on (e.g., "on surface A", "on internal surface B of a multi-layer object") or if it is held by the agent's gripper ('held'). The scene state is $\mathcal{S} = \{s_1, \ldots, s_N\}$.

The set of atomic actions available to the agent is $\mathcal{A} = \{a_0, a_1, a_2, \ldots\}$. Based on the agent's core capabilities, we define an atomic action $a \in \mathcal{A}$ as a parameterized object relocation operation. This conceptually represents a single, high-level step in the interaction, such as moving object o from its current location p to a new valid placement position p'. Formally, executing a feasible action a in state $\mathcal{S}$ transitions the scene to a new state $\mathcal{S}' = a(\mathcal{S})$. These actions primarily modify the position and containment state of the manipulated object, and update the gripper's state.

The fundamental agent-environment interaction (as conceptually illustrated in Fig. 2) follows a discrete-time cycle $\mathcal{S}_t \xrightarrow{a_t} \mathcal{S}_{t+1}$, where at each step t, the agent selects an action a_t from the set of atomic actions feasible in state $\mathcal{S}_t$. The feasibility of actions depends on the current state (e.g., an object must be movable; a target location must be a valid surface that can accommodate the object).

Based on this formal framework, we categorize all possible mobile manipulation tasks by their objective relative to the interaction cycle. **Process-based tasks** explicitly specify a desired sequence of atomic actions $\langle a_1, \ldots, a_n \rangle$ that constitute a feasible trajectory through the state space. For example, the instruction "Move the book from the shelf to the left part of the tabletop" corresponds to a sequence of parameterized object relocation actions. **Outcome-based tasks** specify a desired target scene state $\mathcal{S}_{target}$ without dictating the intermediate actions required to reach it. For example, the instruction "Clear the table" corresponds to a target state where the containment states of relevant objects satisfy specific criteria relative to the table (e.g., no objects are on the table's surface).

This formalization provides a rigorous basis for systematically defining and generating task instances. Sec. 3.2 will detail how to utilize the given scene information to construct a Receptacle-Aware 3D Scene Graph which enables comprehensive retrieval of all objects and receptacles. In Sec. 3.3, we illustrate how to further generate both process-based and outcome-based tasks.

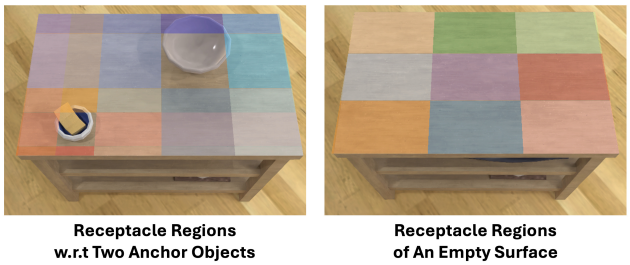

Figure 3. **Visualized Receptacle Regions on a Surface.**

3.2. Receptacle-Aware 3D Scene Graph

We construct a scene graph $\mathbb{S}$ which serves as a structured representation of the scene, encoding both objects and available receptacles. The input scene information includes: object poses, object bounding boxes, and object mesh models (optional for extracting interior receptacles within objects).

The construction process involves two main steps. First, we initialize the scene graph as a structural object spatial relationship tree. This tree is based on spatial containment on surfaces: the root node represents the ground, while all other nodes correspond to scene objects. Parent-child relationships are determined by spatial containment—an object is assigned as a child of another if it rests on its surface. In addition, we will extract the interior surfaces (if any) of an object if its mesh model is available, and provide a more precise record of multi-level object placement during tree initialization. We also compute and record relative positions and distances between objects on the same surface.

After that, we identify the available receptacles within the scene. Building upon the established object relationship tree, we characterize receptacles by segmenting the free space of surfaces. Specifically, we treat each object as an anchor, and segment the free space around it into eight directional receptacle regions, with boundaries extending until they encounter another object or the edge of the supporting surface. For unoccupied surfaces, we employ a 3×3 grid segmentation by default. Visualization of our receptacle segmentation is provided in Fig. 3. Receptacle properties such as location, size, and direction relative to its anchor object or the surface are recorded. We also capture complex cases, such as a single receptacle indexed by multiple objects or the potential for merging adjacent receptacles into larger ones. Finally, all these information are stored as attributes of each node of the scene graph. This representation enables efficient retrieval of any receptacle by querying objects or surfaces. We include more details on the scene graph construction process in Appendix B.

The resulting 3D Scene Graph $\mathbb{S}$ integrates object nodes (encoding properties like pose, bounding box, and relationships) and receptacle information (encoding properties like location, size, and relationships to objects/surfaces). This

unified representation of the scene is crucial for deriving the set of feasible atomic object relocation actions, as detailed in the following section.

3.3. Generating Tasks

Building upon the Receptacle-Aware 3D Scene Graph $\mathbb{S}$ introduced in Sec. 3.2, this subsection details our methodologies for generating comprehensive sets of process- and outcome-based tasks.

Atomic Action Derivation. We first derive the complete set of feasible atomic actions ($\mathcal{A}$) based on $\mathbb{S}$, which explicitly encodes the the properties of every object (location, size, relationships) as well as all available receptacles. As formalized in Sec. 3.1, an atomic action corresponds to a parameterized object relocation operation, such as moving object o from its current location P to a new valid placement position P'. Accordingly, by identifying every object-receptacle pair where the object can be feasibly placed on the receptacle's surface, we enumerate the full set of possible target locations P' for each object. The set of atomic actions $\mathcal{A}$ is thus defined by all currently feasible object-to-receptacle relocation operations within the scene.

Process-based Tasks. As formalized in Sec. 3.1, process-based tasks explicitly specify a desired sequence of atomic actions $\langle a_1, \ldots, a_n \rangle$. Leveraging the agent-environment interaction cycle (Fig. 2) and the derived atomic action set $\mathcal{A}$, we generate diverse process-based tasks by systematically sampling and composing feasible action sequences. Single-step tasks are simply individual actions sampled from $\mathcal{A}$. Multi-step tasks are formed by chaining sequences of actions. The interaction cycle enables this chaining: after executing an action a_t in state $\mathcal{S}_t$, it transitions to $\mathcal{S}_{t+1}$, then we can sample a subsequent action a_{t+1} that is feasible in the new state $\mathcal{S}_{t+1}$.

We compose these sequential actions using logical connectors to form complex process-based task instructions. The most common one for defining a task sequence is THEN (e.g., "Execute a_t THEN execute a_{t+1}"). Other connectors like AND or OR can also be used to generate more diverse task structures, reflecting different types of process specifications. By iteratively sampling and chaining such feasible actions across multiple steps, we construct process-based tasks of varying lengths and complexity.

To further enhance linguistic and spatial diversity, we define and describe the target location for each object relocation action within a sampled sequence using distinct spatial strategies derived from the scene graph. These strategies include: *move to a named surface*, *move to a location around a specific object*, *move to a location with specific direction relative to a specific object*, or *move to a location between two objects*. Furthermore, LLMs can be optionally used to rephrase the entire generated task instruction, increasing linguistic variation while preserving semantics.

Datasets	Num. of Scenes	Num. of Tasks
GenSim [39]	-	100
λ [18]	20	521
M3Bench [47]	119	31,050
ALFRED [31]	120	25,743
Language Rearrangement [35]	1	1,000
Embodied Agent Interface [23]	2	438
EmbodiedBench [43]	4	1,128
ManiTaskGen-RAS-40K (Ours)	3	39,871
ManiTaskGen-RAS (Ours)	3	$+\infty$

Table 1. **Comparison between ManiTaskGen-RAS and Other Existing Datasets.**

Outcome-based Tasks. Generating outcome-based tasks, which define desired target scene states, is more challenging as it requires abstract state descriptions. A naive approach of using VLMs or LLMs directly on the scene information often yields impractical tasks and limited diversity due to model limitations in understanding complex 3D scenes (further discussed in Sec. 4).

To address this, we employ a hybrid approach combining template-based generation and VLM-based filtering. We introduce ManiTaskOT-200, a manually curated outcome-based task template dataset acquired from human-written instructions on diverse scenes, comprising 200 structured templates (details in Appendix C.1). Examples of these templates include:

> "Create a tidy arrangement on [PLATFORM0]."
> "Disorganize [PLATFORM0] to make it messy."
> "Sort all [SUB-OBJECTS00] on [PLATFORM0] by material."

Given a scene, we generate outcome-based tasks by instantiating ManiTaskOT-200 templates with scene-specific objects. To ensure task feasibility, we then employ an ensemble of VLMs to vote on the executability of each generated instruction, filtering out impractical tasks and refining the final set. We provide more details on this process in Appendix C.2.

4. Evaluation of ManiTaskGen Tasks

To evaluate the effectiveness of ManiTaskGen in generating comprehensive and diverse tasks for varying scenes, we apply it to both simulated environments (ReplicaCAD [33], AI2THOR [21]) and real-world scene datasets (SUN-RGBD [32]).

These datasets provide scene information including object poses and bounding boxes, with ReplicaCAD and AI2THOR covering additional object mesh models which enable generating tasks w.r.t object interior surfaces. For

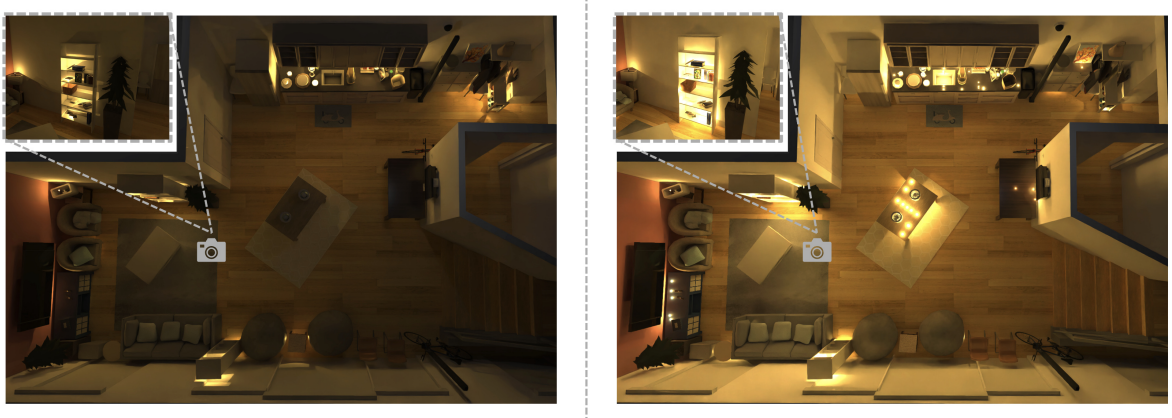

Figure 4. **The "lightmap"s Which Show the Diversity of Generated Tasks.** Each time an object or location is mentioned in a task, we add a highlight at the corresponding position. The brightness distribution reveals that our generated tasks cover more objects and locations.

this evaluation, we selected one representative scene from each dataset and used ManiTaskGen to generate a dataset of tasks, named as ManiTaskGen-RAS. Note that ManiTaskGen's generative process supports arbitrarily complex and length of process-based tasks by sampling from multiple interaction cycles, and instantiating outcome-based task templates with diverse objects also generates a vast number of tasks. Thus, the potential size of the task space is theoretically infinite. Yet, for the purpose of statistical analysis, we curate a finite subset of these tasks, named as ManiTaskGen-RAS-40K, comprising a total of 39,871 tasks. Specifically, for the process-based tasks, we include 39,221 instances comprising single-step pick-and-place tasks (sampled from a single interaction cycle) and two-step tasks (sampled from two consecutive cycles and connected with the logical connector THEN). For the outcome-based tasks, we generated them by instantiating templates part from MANITASKOT-200 and employed an ensemble of VLMs (GPT-4o [1], Gemini-2.5-pro [37], Claude-3.7-sonnet [3]) to vote on their feasibility. This filtering process resulted in a final set of 650 outcome-based task instructions. More details of ManiTaskGen-RAS-40K are provided in Appendix D.1.

We compare our generated datasets with existing embodied decision-making datasets in Tab. 1. Notably, despite using only 3 scenes, ManiTaskGen-RAS-40K contains significantly more tasks than other datasets. Furthermore, our method is scene-agnostic, meaning it can be applied to any given scene, allowing for the incorporation of additional scene data sources to further expand the task set.

Next, we present further results of evaluating the quality of generated tasks from two key aspects : **Validity** and **Diversity**. For a fair comparison, we implement a GPT-based task generation approach as a baseline, referred to as GPTTaskGen. Specifically, we feed each scene's object information along with their images to GPT-4o [1], instructing it to generate the tasks. We apply this baseline method to the same 3 scenes as in ManiTaskGen-RAS-40K to generate 10,000 process-based tasks and 1,000 outcome-based tasks, referred to as GPTTaskGen-RAS. This task set serves as a direct comparison to evaluate the validity and diversity of the tasks produced by our method.

	Process-based Tasks	Outcome-based Tasks
GPTTaskGen-RAS	29.4%	21.1 %
ManiTaskGen-RAS-40K (Ours)	94.0%	86.5 %

Table 2. **Human-Verified Task Validity Rate.**

Validity Assessment. We first evaluate the validity of the generated tasks by conducting human verification on ManiTaskGen-RAS-40K and GPTTaskGen-RAS, with the results reported in Tab. 2. For process-based tasks, although our generation algorithm ensures the target location has sufficient space to accommodate the moved object, some tasks may still be infeasible due to occlusions or obstacles, making the target position difficult to reach or observe. Nevertheless, our results show that most tasks are valid, with significantly higher validity rates compared to the baseline method. For outcome-based tasks, the validity rate is expectedly lower compared to process-based tasks, as it relies on a VLM-based filtering mechanism. Yet our method produces mostly valid tasks.

Diversity Assessment. We proceed to assess the diversity of the generated task set, which reflects how well the tasks cover various scenarios. To compare task diversity, we randomly sample 100 tasks for the same apartment scene of ReplicaCAD from ManiTaskGen-RAS-40K and GPTTaskGen-RAS, and count their object and location coverings. Fig. 4 presents two "lightmap"s that visualizes the distribution of involved objects and locations. Specifically, we light up the centroid of an object or location whenever

it appears in a task. The figure clearly demonstrates that our method generates a wider range of tasks, covering more diverse objects and locations compared to the baseline.

5. Benchmarking and Improving Embodied Decision-Making

In this section, we showcase two important applications of ManiTaskGen: benchmarking and improving embodied decision-making agents. In Sec. 5.1, we design an automatic framework for constructing benchmarks using tasks generated by ManiTaskGen, evaluate the decision-making capabilities of existing VLM agents. In Sec. 5.2, we further propose an inference-time improvement method that utilizes ManiTaskGen tasks and validate its effectiveness through experimental results.

5.1. Benchmarking VLM Agents

To facilitate necessary agent-environment interaction, the benchmark is constructed within an simulator [42]. After loading the given interactive scene and generating the tasks, at each timestep of an episode, the VLM-based agent receives observations from the environment and selects an abstracted action (Sec. 5.1.1) to execute. The environment will be automatically updated and conduct the result judgment (Sec. 5.1.2) if a completion signal is given. Sec. 5.1.3 presents our benchmark results.

5.1.1. Action Space and Test Flow

Action Space. We define a discrete, abstracted action space that the VLM-based agent selects from at each timestep. The scene graph $\mathbb{S}$ introduced in Sec. 3.2 is used to calculate feasible walkable areas around each ground object, which define possible navigation targets, and also to identify platforms within objects and segment them into receptacle regions. The detailed action space includes:

- `go_to(platform_id)`: Navigate to a walkable area around a specific platform of an object.
- `change_view`: Re-navigate to a different walkable area (if any) of the current platform.
- `pick(object_id)`: Grasp the object with the specified ID. The `object_id`s are associated with the tags provided as the visual observation when the agent is located near a platform.
- `show_receptacle(object_id)`: Visualize feasible receptacle regions (tagged by `receptacle_ID`s) associated with an specified object on the current platform.
- `place`: Place the held object at a specified location. We offer two placement modes:
  - `place_r`: Place the object at a random feasible receptacle on the current platform.
  - `place_s([object_id:receptacle_id])`: Place the object within the space defined by the list of receptacle regions associated with specified objects on current platform. If multiple receptacles are provided, the system merges them into a larger placement region if they are connected or overlapped.
- `call_end`: Signals the agent's intention to complete the task and terminates the episode.

Test Flow. An example episode of the benchmark test flow is visualized in Fig. 5. At each timestep, the agent receives multimodal observations from the system, including rendered images (with tags for objects or, after `show_receptacle`, for receptacles), and text information. The system executes the given actions, updates the environment, and provides the next observation. An episode terminates when the agent executes `call_end` or a preset timestep limit is reached. After termination, the benchmark automatically evaluates the episode, following the criteria illustrated in the next section.

5.1.2. Task Difficulty and Evaluation Criteria

Task Difficulty Levels. We classify ManiTaskGen tasks into four difficulty levels. Levels 1 to 3 are process-based tasks, with increasing structural and perceptual complexity. Level 1 comprises single-step pick-and-place tasks involving unique target objects. Level 2 introduces perceptual ambiguity with non-unique target objects requiring disambiguation and additional description, such as moving a red cup when multiple cups are present on the same platform. Level 3 consists of two sub-tasks formed by chaining atomic actions (two one-step pick-and-place tasks from Level 1 or 2) using the logical connector THEN. Level 4 includes all outcome-based tasks.

Evaluation Criteria. For process-based tasks (Levels 1 to 3), the expected final scene state is precisely defined, enabling automatic success verification by comparing the initial and final scene graphs. For each testing episode, we conclude following evaluation metrics: (1) **Success Rate (SR)**; (2) **Intermediate Points (IP)**. For Level 1 & 2 tasks, a successful episode should include the following four substeps, each contributing 25 points of IP: Navigate to the correct starting location; Grasp the correct object; Navigate to the correct destination with the right object; Place the right object in the correct place. IP for Level 3 tasks is computed by averaging points from the two sequential sub-tasks. Regarding Level 4 tasks, which involve abstract descriptions of scene state changes (e.g., "make the desk cleaner"), defining an unbiased and precise success-state scene graph is challenging. Possible evaluation methods include human verification or leveraging VLMs to assess whether the final scene state satisfies the requirements. We leave benchmarking Level 4 tasks in future work.

5.1.3. Benchmarking Results on VLM Agents

We evaluate existing VLM-based embodied agents in the simulated scenes [21, 33] from ManiTaskGen-RAS-40K. Specifically, we randomly sample 1000 tasks for each level

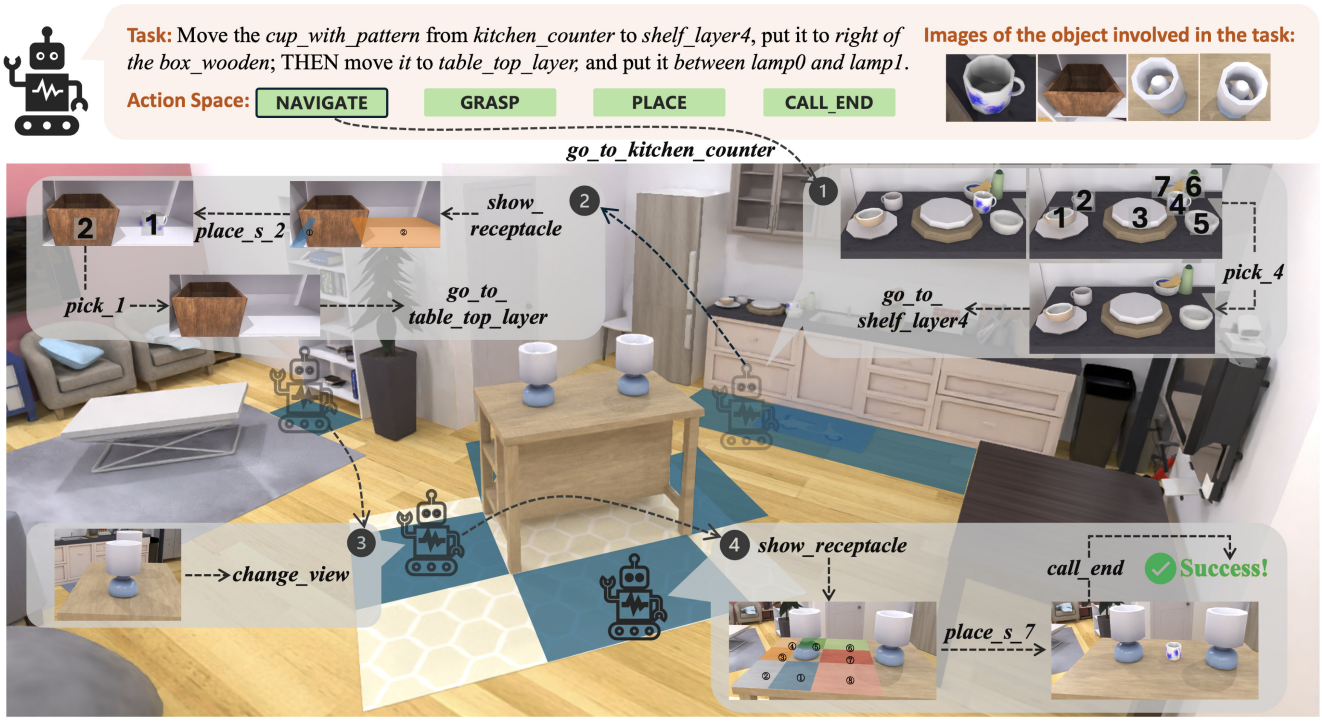

Figure 5. **Visualization of An Example Testing Episode.** The agent is equipped with abstracted navigation (`go_to, change_view`), grasping (`pick`) and placing (`show_receptacle, place`) skills. Blue marks indicate the walkable locations around ground objects involved in this episode.

	Level 1		Level 2		Level 3		Average	
	IP	SR (%)	IP	SR (%)	IP	SR (%)	IP	SR (%)
Human	96.0	82.5	95.0	80.5	95.0	80.0	95.3	81.0
Random	0.7	0.0	0.7	0.0	0.7	0.0	0.7	0.0
GPT-4o[1]	71.6	40.1	42.9	16.3	57.0	8.2	57.2	21.5
GPT-4.1[1]	67.5	36.4	46.4	20.4	54.4	4.2	56.1	20.3
GPT-4.1-mini [1]	67.0	34.0	45.9	22.4	45.3	0.0	52.7	18.8
Gemini-2.5-flash[37]	75.0	40.3	40.8	14.3	56.0	4.1	57.3	19.6
Gemini-2.5-pro [37]	**82.3**	**51.5**	54.1	22.4	**68.9**	**13.3**	**68.4**	**29.1**
Claude-3.7-sonnet [3]	73.9	45.7	**55.6**	**28.6**	54.7	8.9	61.4	27.7
Claude-3.5-haiku [3]	60.0	31.5	40.3	12.2	37.8	2.2	46.0	15.3
Qwen-2.5-VL-72B-Ins [40]	55.1	28.2	37.2	8.2	43.5	4.4	45.3	13.6
Llama-3.3-70B-Vision-Ins [26]	66.1	37.6	42.9	12.2	51.4	0.0	53.5	16.6

Table 3. **Evaluation Results on Existing VLMs.** We cover both proprietary (upper part) and open-source models (lower part). Here, IP refers to Intermediate Points, and SR refers to Success Rate.

of the tasks. A timestep limit of 20 is set for Level 1 & 2 tasks, and 40 for Level 3 tasks. Results are shown in Tab. 3. We observe that all models achieve a low average SR (under 30%), significantly below human-level performance. Performance generally decreases with increasing task difficulty. Longer tasks (Level 3) lead to significantly lower performance compared to single-step tasks, highlighting the substantial challenge posed by the generated tasks. To further analyze failure cases, in Fig. 6, we visualize the sub-step mistake distribution from Gemini-2.5-pro [37] for Level 3 tasks. We observe that wrong placement accounts for the largest proportion of mistakes, suggesting that VLMs' spatial understanding capabilities may serve as a primary bottleneck for mobile manipulation tasks. Furthermore, analysis of execution traces reveals that while the agent performs well in navigation in early stages of the episode (e.g., navigating to the first target object), it makes more navigation mistakes in later steps, indicating that longer-horizon tasks pose a greater challenge to the VLM agent's decision-making capabilities.

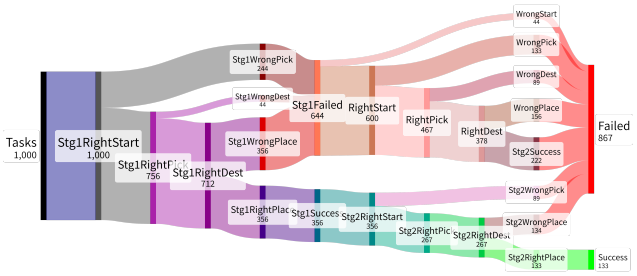

Figure 6. **Success and Failure Modes** from Gemini-2.5-pro in each stage of Level 3 tasks.

Model Name	IP (Before)	IP (After)	SR (Before)	SR (After)
GPT-4.1[1]	75.5	**79.4**	38%	**49%**
Gemini-2.5-flash[37]	79.4	**81.7**	36%	**58%**
Claude-3.5-haiku [3]	68.9	**79.4**	36%	**51%**

Table 4. **Improvement of Agent Performance.**

5.2. Improving VLM Agents

ManiTaskGen tasks can not only be used for benchmarking VLMs, but also for optimizing VLM-based agents. There are two prominent directions for improvement: one leverages precisely labeled task trajectories for supervised fine-tuning (SFT) [25, 41], and the other is reinforcement fine-tuning (RFT) [30, 46], utilizing feedback of task execution process for unsupervised optimization. Given that ManiTaskGen enables automated evaluation of the final results as well as intermediate steps (Sec. 5.1), we adopt an inference-time RFT policy to enhance agent capabilities inspired by Reflexion [30] and ReAct [45]. Specifically, we design a self-reflection model that processes the evaluation results of each episode to generate a verbal summary. This summary is then stored in a long-term memory and used as part of the input for the agent when attempting the task in future episodes. We present details of our optimization process in Sec. 5.2.1, and experimental results in Sec. 5.2.2.

5.2.1. Optimization based on Self-Reflection

Given a trial episode and its evaluation results (including final success judgment and intermediate step evaluations), a rule-based self-reflection model automatically generates the following summary:

- The task goal, the history of the trial episode, and whether the task was ultimately successful.
- If the task was not successful, a summary indicating which intermediate step goals were achieved and the point in the action history up to which they were successfully completed. As described in Sec. 5.1.2, these correct intermediate steps include: navigate to the starting location, grasp the object, navigate to the destination location, and place the object .
- For the intermediate goals underachieved, what are the next suggested actions.

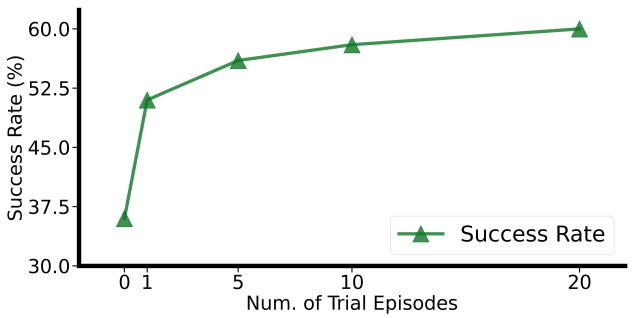

Figure 7. **Improvement of Gemini-2.5-flash Agent.** We find that more trial episodes continuously contribute to the performance.

- Corresponding observations in each part of the summary.

We maintain a long-term memory to store these verbal summaries from multiple trial episodes. The optimization process is iterative. After each trial, the corresponding self-reflection updates the memory, which is used as part of the input for the agent when attempting subsequent trials.

5.2.2. Improvement Results

To ensure no data leakage between optimization and deployment, we select tasks from the ReplicaCAD [33] scene of ManiTaskGen-RAS as source trial episodes, and subsequently tested in the AI2THOR [21] scene. Tab. 4 summarizes the performance improvements of agents after optimization. We used 10 trial episodes for optimization, then conducted evaluation on 100 randomly sampled test tasks, which are all Level 1 tasks. As shown in the table, our optimization significantly improved both the task execution success rate and the Intermediate Points score. We further investigated the impact of the number of trial episodes on final agent performance. Results are presented in Fig. 7. Preliminary results show that agent performance continuously improves with an increasing number of self-reflection trial episodes. Similar findings have been observed in previous work [30]. Therefore, these findings further underscore the significance of ManiTaskGen. Because it can generate and evaluate tasks in arbitrary scenes, thus providing abundant resources for agent improvement.

6. Conclusion

In this paper, we introduce ManiTaskGen, an automated method for task generation for any given scene. ManiTaskGen can generate a comprehensive set of long-horizon mobile manipulation tasks, covering both process-based and outcome-based tasks, thereby providing a diverse set of testing scenarios and improving resources for embodied decision-making agents. Our experiments demonstrate the validity and diversity of the generated tasks, while also showcasing their practical usability by benchmarking and improving the decision-making capabilities of existing VLM-based agents.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, and Shyamal Anadkat et al. Gpt-4 technical report, 2024. 5, 7, 8

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

[3] Anthropic. About claude models. https://docs.anthropic.com/en/docs/about-claude/models, 2024. Accessed: 2024-09-03. 5, 7, 8

[4] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023. 1

[5] Dhruv Batra, Angel Chang, Sonia Chernova, Andrew Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, Manolis Savva, and Hao Su. Rearrangement: A Challenge for Embodied AI. *arXiv preprint*, 2020. 1, 2

[6] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control, 2024. *URL https://arxiv. org/abs/2410.24164*, 2024. 1

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. 1

[8] Zhili Cheng, Yuge Tu, Ran Li, Shiqi Dai, Jinyi Hu, Shengding Hu, Jiahao Li, Yang Shi, Tianyu Yu, Weize Chen, et al. Embodiedeval: Evaluate multimodal llms as embodied agents. *arXiv preprint arXiv:2501.11858*, 2025. 2

[9] Jae-Woo Choi, Youngwoo Yoon, Hyobin Ong, Jaehong Kim, and Minsu Jang. Lota-bench: Benchmarking language-oriented task planners for embodied agents. *arXiv preprint arXiv:2402.08178*, 2024. 2

[10] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020. 2

[11] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. *arXiv preprint*, 2023. 1

[12] Kuan Fang, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Adaptive procedural task generation for hard-exploration problems. *arXiv preprint arXiv:2007.00350*, 2020. 2

[13] Jensen Gao, Bidipta Sarkar, Fei Xia, Ted Xiao, Jiajun Wu, Brian Ichter, Anirudha Majumdar, and Dorsa Sadigh. Physically grounded vision-language models for robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12462–12469. IEEE, 2024. 1

[14] Jiayuan Gu, Devendra Singh Chaplot, Hao Su, and Jitendra Malik. Multi-skill mobile manipulation for object rearrangement. *arXiv preprint arXiv:2209.02778*, 2022. 1, 2

[15] Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jianguang Lou, Qingwei Lin, Ping Luo, Saravan Rajmohan, and Dongmei Zhang. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. *arXiv preprint arXiv:2408.00764*, 2024. 2

[16] Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. *arXiv preprint arXiv:2311.12871*, 2023. 1

[17] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022. 1

[18] Ahmed Jaafar, Shreyas Sundara Raman, Yichen Wei, Sudarshan Harithas, Sofia Juliani, Anneke Wernerfelt, Benedict Quartey, Ifrah Idrees, Jason Xinyu Liu, and Stefanie Tellex. λ: A benchmark for data-efficiency in long-horizon indoor mobile manipulation robotics, 2025. 4

[19] Emily Jin, Jiaheng Hu, Zhuoyi Huang, Ruohan Zhang, Jiajun Wu, Li Fei-Fei, and Roberto Martín-Martín. Mini-behavior: A procedurally generated benchmark for long-horizon decision-making in embodied ai. *arXiv preprint arXiv:2310.01824*, 2023. 2

[20] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 1

[21] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. 2, 4, 6, 8

[22] 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. 2

[23] Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Erran Li Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37: 100428–100534, 2025. 2, 4

[24] Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, et al. Visualagentbench: Towards large multimodal models as visual foundation agents. *arXiv preprint arXiv:2408.06327*, 2024. 2

[25] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing

data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR, 2023. 8

[26] Meta. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models, 2024. Accessed: 2025-02-15. 1, 7

[27] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021. 2

[28] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024. 2

[29] Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020. 2

[30] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023. 2, 8

[31] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020. 2, 4

[32] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015. 2, 4

[33] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2, 4, 6, 8

[34] Andrew Szot, Karmesh Yadav, Alex Clegg, Vincent-Pierre Berges, Aaron Gokaslan, Angel Chang, Manolis Savva, Zsolt Kira, and Dhruv Batra. Habitat rearrangement challenge 2022. https://aihabitat.org/challenge/2022_rearrange, 2022. 2

[35] Andrew Szot, Max Schwarzer, Harsh Agrawal, Bogdan Mazoure, Rin Metcalf, Walter Talbott, Natalie Mackraz, R Devon Hjelm, and Alexander T Toshev. Large language models as generalizable policies for embodied tasks. In *The Twelfth International Conference on Learning Representations*, 2023. 1, 4

[36] Andrew Szot, Bogdan Mazoure, Omar Attia, Aleksei Timofeev, Harsh Agrawal, Devon Hjelm, Zhe Gan, Zsolt Kira, and Alexander Toshev. From multimodal llms to generalist embodied agents: Methods and lessons. *arXiv preprint arXiv:2412.08442*, 2024. 1

[37] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, and et al. Anja Hauth. Gemini: A family of highly capable multimodal models, 2024. 5, 7, 8

[38] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. 1

[39] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. *arXiv preprint arXiv:2310.01361*, 2023. 2, 4

[40] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 7

[41] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021. 8

[42] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020. 6

[43] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents, 2025. 2, 4

[44] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025. 2

[45] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. 2, 8

[46] Simon Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Peter Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *Advances in neural information processing systems*, 37:110935–110971, 2024. 8

[47] Zeyu Zhang, Sixu Yan, Muzhi Han, Zaijin Wang, Xinggang Wang, Song-Chun Zhu, and Hangxin Liu. M3bench: Benchmarking whole-body motion generation for mobile manipu-

lation in 3d scenes. *arXiv preprint arXiv:2410.06678*, 2024.
2, 4