MOKA: Open-Vocabulary Robotic Manipulation through Mark-Based Visual Prompting

Fangchen Liu* Kuan Fang* Pieter Abbeel Sergey Levine
Berkeley AI Research, UC Berkeley

https://moka-manipulation.github.io

Abstract—**Open-vocabulary generalization requires robotic systems to perform tasks involving complex and diverse environments and task goals. While the recent advances in vision language models (VLMs) present unprecedented opportunities to solve unseen problems, how to utilize their emergent capabilities to control robots in the physical world remains an open question. In this paper, we present Marking Open-vocabulary Keypoint Affordances (MOKA), an approach that employs VLMs to solve robotic manipulation tasks specified by free-form language descriptions. At the heart of our approach is a compact point-based representation of affordance and motion that bridges the VLM's predictions on RGB images and the robot's motions in the physical world. By prompting a VLM pre-trained on Internet-scale data, our approach predicts the affordances and generates the corresponding motions by leveraging the concept understanding and commonsense knowledge from broad sources. To scaffold the VLM's reasoning in zero-shot, we propose a visual prompting technique that annotates marks on the images, converting the prediction of keypoints and waypoints into a series of visual question answering problems that are feasible for the VLM to solve. Using the robot experiences collected in this way, we further investigate ways to bootstrap the performance through in-context learning and policy distillation. We evaluate and analyze MOKA's performance on a variety of manipulation tasks specified by free-form language descriptions, such as tool use, deformable body manipulation, and object rearrangement.**

I. Introduction

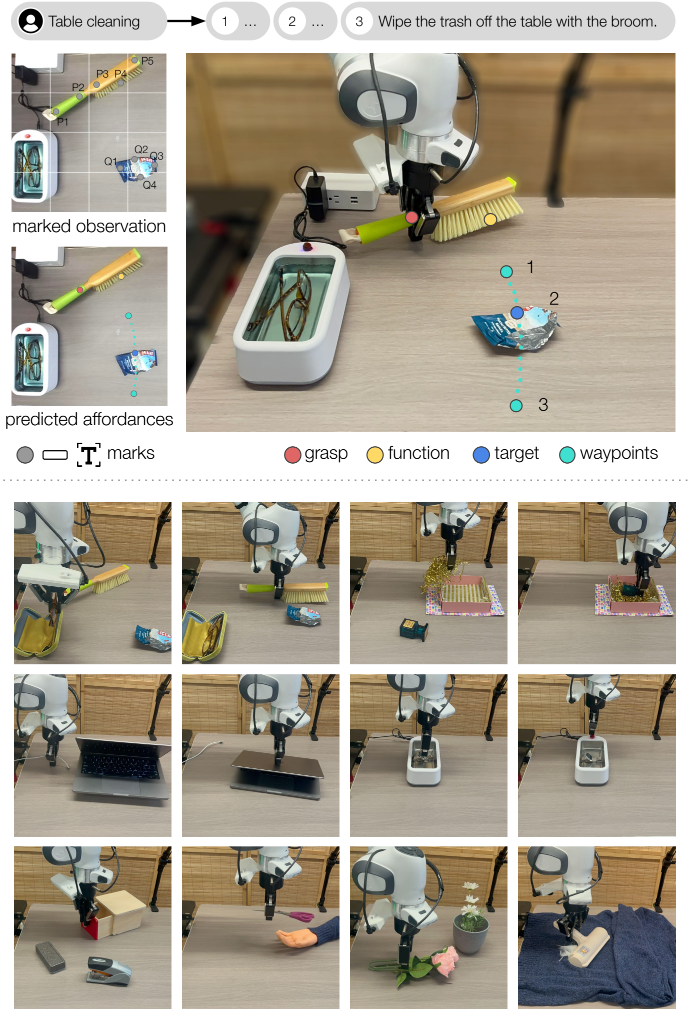


Fig. 1: To solve manipulation tasks with unseen objects and goals, MOKA employs a VLM to generate motions through a point-based affordance representation (plotted as colorful dots on the images). By annotating marks (e.g., candidate points, grids, and captions) on the observed 2D image, MOKA converts the motion generation problem into a series of visual question-answering problems that the VLM can solve.

The pursuit of open-vocabulary generalization poses a major challenge for robotic systems: Solving tasks in unseen environments given new user commands necessitate methods that can deal with the vast diversity and complexity of the physical world. An appealing prospect for handling this challenge is to employ large pretrained models by encapsulating extensive prior knowledge from broad data and bringing it to bear on novel problems. Recent advances in large language models (LLMs) and vision-language models (VLMs) provide particularly promising tools in this regard, with their emergent and fast-growing conceptual understanding, commonsense knowledge, and reasoning abilities [8, 39, 40, 4, 7, 1, 41, 22, 42, 23]. However, existing large models pre-trained on Internet-scale data still lack the capabilities to understand 3D space, contact physics, and robotic control, not to mention the knowledge about the embodiment and environment dynamics in each specific scenario, creating a large gap between the promising trend in computer vision and natural language processing and applying them to robotics. It remains an open question how such tools can guide a robotic system to solve manipulation tasks by interacting with the physical world.

Recently, a growing body of research has been dedicated to utilizing pre-trained large-scale models for robotic control. By incorporating broad knowledge, these approaches directly

*: Equal contribution. Correspondance to {fangchenliu, kuanfang}@berkeley.edu

prompt or fine-tune the large models to generate plans [2, 16, 15, 5], rewards [28, 20, 30, 56], codes [25, 45, 49], etc. Despite the encouraging results they have demonstrated, these approaches are subject to notable limitations. Since the advances in LLMs precede VLMs, many previous approaches first process the raw sensory inputs to obtain the language description of the environment and then query LLMs to perform reasoning and planning in the language domain. However, relying solely on high-level language descriptions may overlook the nuanced visual details of environments and objects, which are vital for accurately completing tasks. In addition, existing approaches usually require non-trivial effort in designing in-context examples [17, 25, 45] to ensure LLMs can produce desired predictions on similar tasks. As a result, the tasks that can be solved by these approaches are largely constrained by such manual efforts.

In this work, we study how to effectively endow robots the ability to solve novel manipulation tasks specified by free-form language instructions using VLMs. Our key insight is to find an intermediate affordance representation that connects the VLM's prediction on images with the robot's motion in the physical world. This affordance representation should satisfy two critical requirements. First, it should be feasible for the VLM to predict given the visual observation of the environment and task description. Second, it should compactly capture the information that well characterizes the important properties of the robot's motion, such that it can be easily executed on the robot.

To this end, we propose Marking Open-vocabulary Keypoint Affordances (MOKA), an approach that employs VLMs for robotic manipulation through mark-based visual prompting. As shown in Fig. 1, MOKA leverages a compact affordance representation consisting of a set of keypoints and waypoints, defined on open sets of objects and tasks. This point-based affordance representation is then used to specify the desired motion for the robot to solve the task. To generate the motions given the free-form language descriptions, MOKA uses hierarchical visual prompting to convert the affordance reasoning problem into a series of visual question answering problems.

Drawing inspirations from recent advances in visual question-answering [51], we use mark-based visual prompting to enable the VLM to attend to the important visual cues in the observation image and further simplify the point generating problem into multiple choice questions. As shown in the top-left part of Fig. 1, we plot the keypoints on the image, and query the VLM to select the keypoints that result in the desired motion. The predicted keypoints and waypoints are used for specifying a trajectory through a waypoint-following motion, which can represent a wide range of manipulation skills such as picking, placing, pressing, tool-use, etc.

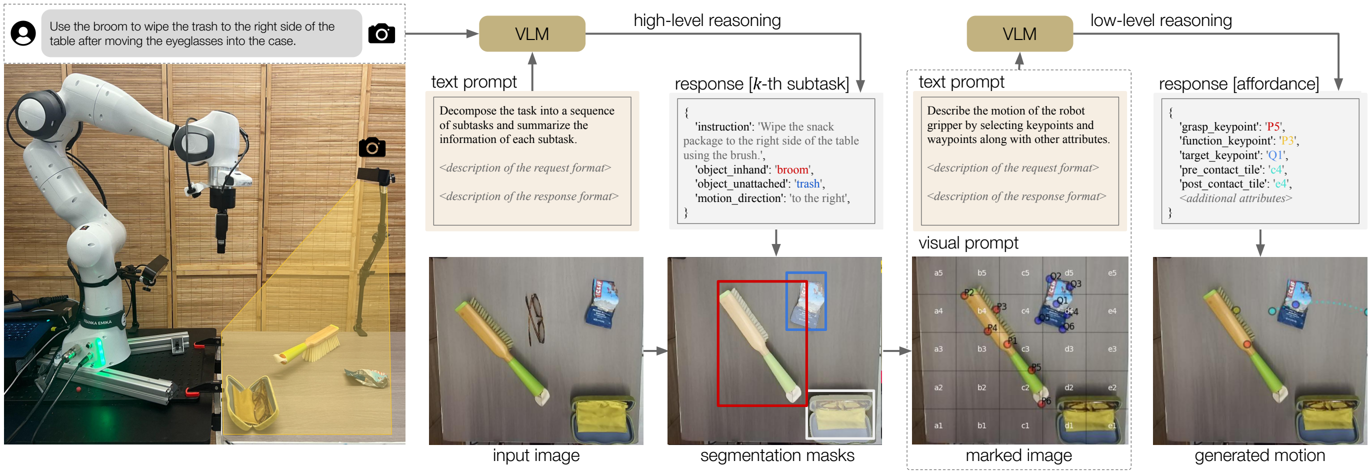
We demonstrate the effectiveness of MOKA on a variety of manipulation tasks, with robustness across the variations of instructions, objects, and scenes. We further use MOKA to collect successful trajectories in each task. The collected trajectories can be used as in-context examples to further bootstrap the performance of VLM, and can also be leveraged as demonstrations to train a better student policy. Our experiments show that MOKA can achieve state-of-the-art performance on our proposed evaluation tasks in a zero-shot manner, with consistent improvements using clean and intuitive in-context examples. To summarize, our contributions are:

- We introduce a point-based affordance representation that bridges the VLM's prediction on RGB images and the robot's motion in the physical world.
- We propose a mark-based visual prompting approach that converts the affordance reasoning problem into a series of visual question answering problems.
- We demonstrate that the resultant approach, MOKA, can effectively generate motions to solve diverse open-vocabulary manipulation tasks.

II. Related Work

Large language models and vision-language models. Recent developments in various domains and applications have been greatly influenced by the substantial progress achieved through large language models (LLMs) and vision language models (VLMs) [8, 39, 40, 4, 7, 1, 41, 22, 42, 23]. While these models can already solve various tasks in a zero-shot manner [1], well-designed prompts still serve an important role in further eliciting more advanced capabilities. As demonstrated by Brown et al. [4], few-shot prompting can match or surpass the performance of fine-tuning on LLMs. Additionally, other prompting techniques [50, 57, 19, 55] have been proposed to improve LLMs' reasoning capabilities. Although these methodologies may initially appear enigmatic, they have been empirically validated to consistently demonstrate scalability across various models [4, 7, 1, 54]. Apart from the language prompts, recent vision models [18, 58, 21] are capable of supporting visual prompts, including points, boxes, masks, and texts. Such visual prompts exhibit greater diversity in both form and content, harnessing vision-language models for various application scenarios like perception [53], image editing [44] and reasoning [51].

To build an autonomous agent capable of making decisions in an unstructured environment, the incorporation of robust visual reasoning capabilities becomes imperative. Although the current generation of VLMs cannot seamlessly engage in zero-shot reasoning, they can still be effectively harnessed across a diverse spectrum of tasks. Prior work SoM [51] draws visual marks on the objects in an image using numbers and segmentation masks, and demonstrates the mark-based visual prompting scheme unleashes the reasoning capabilities of GPT-4V, such as object counting and inferring spatial relationship. Different from SoM [51], we want to leverage GPT-4V for open-vocabulary robot manipulation. We represent each manipulation phase with a set of affordance points and motion waypoints. Instead of using object segmentation masks, we use keypoints and grid cells as visual prompts as shown in Fig. 1, and then query GPT-4V to choose the affordance points and motion waypoints from the visual marks, followed by executable point-based motion plans.



Use the broom to wipe the trash to the right side of the table after moving the eyeglasses into the case.

Decompose the task into a sequence of subtasks and summarize the information of each subtask.

<description of the request format>

<description of the response format>

```
{
  'instruction': 'Wipe the snack package to the right side of the table using the brush.',
  'object_inhand': 'broom',
  'object_unattached': 'trash',
  'motion_direction': 'to the right',
}
```

Describe the motion of the robot gripper by selecting keypoints and waypoints along with other attributes.

<description of the request format>

<description of the response format>

```
{
  'grasp_keypoint': 'P5',
  'function_keypoint': 'P3',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'c4',
  'post_contact_tile': 'e4',
  <additional attributes>
}
```

Fig. 2: **Overview of MOKA.** We propose a hierarchical approach to prompt the VLM to perform affordance reasoning. On the high-level, we query the VLM to decompose the free-form language description of the task into a sequence of subtasks and summarize the subtask information. On the low-level, the VLM is prompted to produce the keypoints and additional attributes for the affordance representation defined in Sec. IV-A.

Foundation models in robotics. Building upon the successes of large language models, the field of robotics is currently witnessing a rising interest in utilizing LLMs in various application scenarios. LLMs is capable of generating high-level task plan in natural language [2, 16, 15, 5], executable programs [25, 45, 49] or value function [17, 26] for low-level behaviours, environmental reward and feedback for reinforcement learning [28, 20, 30, 56]. However, these approaches necessitate the conversion of both the task and the observations into the textual form. While this can be easily accomplished in simulated environments with ground-truth object state, tasks in real world require the utilization of robust and precise perception modules. To perform open-ended navigation and manipulation in real world [46, 5, 10], open-vocabulary vision foundation models [18, 58, 21, 27, 34, 24] are often used to extract visual scene information before converting the observation to textual form. However, the process of converting an image into text may result in the loss of important details, such as shape and geometric information. With the recent advancements in vision language models (VLMs), it is now conceivable that the role of state estimation combined with language models can be replaced by a single, powerful VLM, such as GPT-4V, which is leveraged by MOKA. Hu et al. [14] utilizes GPT-4V to get semantic language plans and convert each language plan to a set of pre-defined low-level skills. Different from Hu et al. [14], we represent manipulation affordance and motion using a set of points, and query GPT-4V to select the corresponding points (e.g. grasping point, function point) in each task, which can be directly translated to low-level point-based motions. While preserving the visual reasoning capabilities of VLMs, MOKA provides a framework with more general and flexible low-level motion, which doesn't require prior knowledge about a specific task compared to pre-defined skills.

Affordance reasoning for robotic control. The psychologist James J. Gibson, along with Eleanor J. Gibson, introduced the concept of an affordance [11], which refers to the ability to perform a certain action with an object in a given environment. Much of the research related to affordances has concentrated on predicting how to interact with objects, as demonstrated by [47, 12, 3]. In the field of robot manipulation, keypoints are often used to provide compact information about the environment and the objects [6, 9, 48, 31, 33, 32, 38], representing the affordance in a structured way. Among these works, the most related one to ours is KETO [38], which predicts affordance and functional keypoints on the tool objects by learning from interactions with a trajectory optimization formulation as in Manuelli et al. [32]. In contrast to KETO [38], our keypoints selection procedure doesn't require any model training. We design an automated process to annotate keypoints as visual marks on a 2D image, and leverage the broad knowledge from GPT-4V to select affordance and motion keypoints for manipulation.

III. Problem Statement

Our goal is to enable robots to perform manipulation tasks involving unseen objects and goals. Each task is described by a free-form language description l. To achieve success, the robot needs to interact with the environment across one or multiple stages. An example task is shown in Figure 1, where the robot is commanded by "Swipe the snack package off the table. Be careful with the eyeglasses on the table.".

We refer to the physical interaction at each stage as a *subtask*. These subtasks can include interactions with objects in hand (e.g., lifting up an object, opening a drawer, turning the faucet), interactions with environmental objects unattached to the robot (e.g., pushing an obstacle, pressing a button), and tool use which involves grasping a tool object to make contact with another object (e.g., poking with a stick, sweeping with a foam, cutting with a knife). In this example, the robot needs to perform two subtasks: First, put the eyeglasses, which are between the broom and the snack package, back into the case; and second, use the broom to sweep the trash. Since we do not assume the sequence of subtasks is given beforehand, the robot needs to decompose the task into a sequence of feasible subtasks based on the free-form language description l and control the robot to solve each subtask.

We consider a table-top manipulation setting that involves a robotic arm and RGBD cameras, as shown in Figure 2. At each time step t, an observation s_t is received from the environment and an action a_t is chosen to command the robot. The observation s_t consists of the RGBD images captured by the camera sensors as well as the proprioception information. The action a_t is defined in the Cartesian space.

In this paper, we aim to solve such tasks by leveraging vision-language models (VLMs) with an effective visual prompting strategy. We use $\mathcal{M}$ to denote the VLM $\mathcal{M}$ which can take varying numbers of language and visual inputs in a specific order, and generate text responses accordingly. The responses of $\mathcal{M}$ can be controlled by designing different input prompts, which specify information such as the problem descriptions, the input and output formats, and in-context examples [17, 25, 28, 45]. Specifically, we designed text and visual prompts to enable $\mathcal{M}$ to generate desired text responses in a structured format, which can be easily parsed for downstream usage.

IV. Marking Open-vocabulary Keypoint Affordances

We propose Marking Open-vocabulary Keypoint Affordances (MOKA), an approach that leverages the emergent reasoning capability of Vision-Language Models (VLMs) to guide a low-level motion generator to solve unseen tasks specified by free-form language instructions. As shown in Fig. 2, MOKA uses a point-based affordance representation to connect the VLM's prediction on 2D images with the robot's motion in the physical world. To effectively employ the pre-trained VLM, MOKA converts the problem of affordance reasoning into a series of visual question-answering problems that are feasible for VLMs to solve.

In this section, we will start by introducing our point-based affordance representations and how they are used for generating motions. Then, we will describe a hierarchical framework that effectively prompts VLMs for affordance reasoning. After this, we will explain a novel visual prompting approach that lifts the VLM's reasoning capabilities by annotating a set of marks (including candidate points, grids, and labels) on 2D images. Lastly, we will investigate two ways of bootstrapping the performance of our approach through interactions collected in the physical world.

A. Motion with Point-based Affordances

To leverage VLMs for solving open-vocabulary manipulation tasks, there needs to be an interface that connects the inputs and outputs of the VLM and the motions performed by the robot. To achieve this goal, we design an affordance representation defined on 2D images. Produced as the end result by the VLM, the affordance representation specifies the desired motion.

By extending the definitions in Manuelli et al. [32] and Qin et al. [38], we design a point-based affordance representation for a wide range of manipulation tasks. Instead of separately devising motion primitives for different pre-defined skills, we use a unified set of keypoints and waypoints to specify the motion. These points are predicted by VLMs on 2D images and converted to poses in the $\mathbf{SE(3)}$ space. Then a smooth motion trajectory is generated based on these poses. To perform the task, the robot gripper interacts with the environment by following the generated motion trajectory.

We specify the robot's motion in an object-centric manner as shown in Fig. 2. Following the discussion in Sec. III, we would like this representation to be applicable to different types of interactions with objects in the environment. Therefore, we consider two types of objects, $o_{\text{in-hand}}$ (e.g., the broom) and $o_{\text{unattached}}$ (e.g., the trash), and specify the motion with a grasping phase and a manipulation phase. In the grasping phase, the robot reaches and grasps an object $o_{\text{in-hand}}$ from the environment. Then in the manipulation phase, the robot performs a motion and makes contact with another object $o_{\text{unattached}}$, either directly or using $o_{\text{in-hand}}$ as a tool. In some scenarios, only one of these two types of objects is interacted with by the robot, either $o_{\text{in-hand}}$ (e.g., unplugging a cable, opening a drawer) or $o_{\text{unattached}}$ (e.g., pressing a button), and one of the two phases can be skipped accordingly.

We now describe the definition of the keypoints and waypoints as well as how they are used to specify the motions in both phases. These points are illustrated in Fig. 2 and more examples can be found in Sec. V-C. Following the practice of Manuelli et al. [32] and Qin et al. [38], we use the **grasping keypoint** x_{grasp} to specify the position on $o_{\text{in-hand}}$ where the robot gripper should hold the object. If $o_{\text{in-hand}}$ is not involved in a task, the grasping phase will be skipped. For the manipulation phase, the robot's gripper follows a motion trajectory specified by an additional set of points. The **function keypoint** x_{function} specifies the part of $o_{\text{in-hand}}$ that will make contact with $o_{\text{unattached}}$ in the manipulation phase. If $o_{\text{in-hand}}$ is not specified, x_{function} will be on the robot gripper and the contact will directly be made between the robot and $o_{\text{unattached}}$. Correspondingly, the **target keypoint** x_{target} is the part of $o_{\text{unattached}}$ that will be contacted by x_{function} during the manipulation phase. We also introduce the **pre-contact waypoints** $x_{\text{pre-contact}}$ and the **post-contact waypoints** $x_{\text{post-contact}}$ defined in free space, which dictates the manipulation motion along with the keypoints defined on the objects.

During the manipulation phase, the robot moves the gripper such that x_{function} follows the path sequentially connecting the $x_{\text{pre-contact}}$, x_{target}, and $x_{\text{post-contact}}$. Besides following the path, we also require the robot gripper to follow the specified **grasping orientation** R_{grasp} and **manipulation orientation** $R_{\text{manipulate}}$ during the two phases respectively. To better illustrate the design of our point-based motion, we provide examples of the predicted point specifications and the resultant motions from our experiments in Appendix and Sec. V-C. In MOKA, this set of keypoints and **additional attributes** (described in Sec. IV-C) are summarized in a dictionary as the affordance representation (see Fig. 2).

B. Affordance Reasoning with Vision-Language Models

To predict the defined affordance representations, we employ the VLM $\mathcal{M}(\cdot)$ (defined in Sec. III), which is pre-trained on Internet-scale data for solving general visual question answering (VQA) problems. Using a hierarchical prompting framework as shown in Fig. 2, MOKA converts this affordance reasoning problem into a series of VQA problems that are solvable by the pre-trained VLM.

The hierarchical prompting framework takes as input the free-form language description l of the task and an RGB image observation of the environment s_t. MOKA examines the initial observation s_0 and decomposes the task l into a sequence of subtasks using the VLM. For each of the subtasks, the VLM is asked to provide the summary of the subtask instruction, the description of the corresponding $o_{\text{in-hand}}$, $o_{\text{unattached}}$, as well as the description of the motion (e.g., ”from left to right”). On the low level, given the response from the high-level reasoning and the visual observation $s_{t(k)}$ at the beginning of k-th subtask (at the time step $t(k)$), the VLM is queried again with a different prompt to produce the affordance representation defined in Sec. IV-A. In the remainder of this section, we will describe the input formats, output formats, and prompt designs that we use to instantiate this method. Further details, including the complete prompts, can be found in the Appendix B.

High-level reasoning. Given the initial observation s_0 and the language description l, we first query the VLM $\mathcal{M}$ with the language prompt p_{task} to produce the response y_{high}:

$$y_{\text{high}} = \mathcal{M}([p_{\text{high}}, l, s_0]). \tag{1}$$

The representation y_{high} is a string that contains structured information for the K subtasks that the VLM infers are needed to solve the task. We design the prompt so as to require the VLM to produce y_{task} as a list of dictionaries. As shown in Fig. 2, each dictionary contains the language description of a subtask (e.g., *“Wipe the snack package to the right side of the table using the broom.”*), as well as detailed information to facilitate motion generation, including the description of $o_{\text{in-hand}}$ (e.g., *“broom”*), the object name of $o_{\text{unattached}}$ (e.g., *“snack package”*), and the description of the motion (e.g., *“from left to right”*). This high-level plan will be used as an intermediate result for producing the detailed affordance representation through the low-level reasoning with the VLM.

Low-level reasoning. Next, we prompt the VLM once again to produce the affordance representation defined in Sec. IV-A as y^k_{low}, conditioning on the high-level representation y_{high} and the visual observation $s_{t(k)}$ at the beginning of the k-th subtask. Instead of directly predicting 3D coordinates on 2D images, which is challenging and even ill-defined, we query the VLM to output 2D coordinates on the images and deproject them back to the 3D space. The three keypoints $x_{\text{grasp}}, x_{\text{function}}$ and x_{target} are defined on the object surface, and thus we can compute the 3D coordinates using the corresponding depth value of the 2D location based on the RGB image and camera parameters. For the waypoints in free space, we query the VLM to predict the desired height in text. To produce such an affordance representation y^k_{motion}, we query the VLM again by

$$y^k_{\text{low}} = \mathcal{M}([p_{\text{low}}, y^k_{\text{task}}, f(s_{t(k)})]), \tag{2}$$

where y^k_{high} is the substring corresponding to the k-th subtask extracted from y_{high}, and $f(\cdot)$ is a function that process the raw visual observation $s_{t(k)}$. We will explain the motivation and detailed implementation of $f(\cdot)$ in the next section and Appendix B. Through our ablation study in Appendix C, the hierarchical prompting strategy is essential for VLM to successfully perform the affordance reasoning for solving the tasks.

C. Mark-Based Visual Prompting

To perform the low-level reasoning mentioned in the previous section, we need the VLM to generate keypoints and waypoints on 2D images in order to execute a specific motion for a subtask. Since VLMs are better at multiple-choice problems than directly producing continuous-valued locations, we employ a mark-based visual prompting strategy to extract the desired output from VLMs, which we will describe in this subsection.

Inspired by Yang et al. [52], MOKA uses a set of marks as visual prompts to enable VLM to apply its reasoning capability to predict the point-based affordance representation as shown in Fig. 2. Consisting of dots, grids, and text notations annotated on the image observation, these marks play an important role in the reasoning process. Proposed by open-vocabulary object detection and segmentation algorithms, these marks facilitate visual reasoning by encouraging the VLM to attend to the target objects and other task-relevant information in the image. We annotate marks as candidate parts and regions for the VLM to choose the points from, converting the original problem of directly generating coordinates into multiple-choice questions, which is usually more tractable for existing VLMs.

To select keypoints, which are defined on the in-hand object $o_{\text{in-hand}}$ and the unattached object $o_{\text{unattached}}$ suggested by the high-level reasoning in Sec. IV-B, we propose and plot candidate keypoints on these objects. Given the names of $o_{\text{in-hand}}$ and $o_{\text{unattached}}$, we first segment these two objects using GroundedSAM [43], which combines GroundingDINO [27] and SAM [58] to extract segmentation masks of objects specified by a text prompt. After we obtain the segmentation masks of $o_{\text{in-hand}}$ and $o_{\text{unattached}}$, we perform farthest point sampling [37] on the object contour to obtain K boundary points. Together with its geometric center, and overlay the $K + 1$ candidate keypoints on each object. Each candidate keypoint is assigned an index, which is annotated next to it as a reference. To avoid confusion, we use different colors for candidate keypoints on $o_{\text{in-hand}}$ and $o_{\text{unattached}}$ and use the caption in the format of P_i and Q_j respectively, where i and j are integers. More implementation details can be found in Appendix B-B.

Selecting waypoints in free space involves searching over a much larger region. Instead of directly sampling points in the entire workspace, we divide the observed RGB image into an $M \times n$ grid, where m and n are integers. Both m and n are set to 5 for our evaluation tasks. The VLM is prompted to choose the tiles in which the pre-contact and post-contact keypoints are supposed to locate in and then the exact waypoints are sampled uniformly within the tile. For this purpose, we overlay the grid along with the name of each tile on the image. The tile names follow chess notation, which uses letters to specify the columns and integers for the rows.

D. Zero-shot Execution in Real World

After we obtain the selected keypoints and waypoints from the VLM, we will convert the affordance to an executable motion on a real robot. To achieve this goal, we need to lift up all the points from the 2D image to 6D Cartesian space.

For the keypoints defined on the object, we can directly take the corresponding point on the registered depth image and transform it into the robot's frame. For the waypoints that are selected from the free space, since it's not attached to any objects, its height needs to be specified in order to be deprojected to 3D space. For this purpose, we also ask the VLM to determine the height of waypoints. Here We only consider the cases where the waypoints are at the same height as the target point (e.g. pushing), or above the target point (e.g. placing) in most common tabletop manipulation scenarios. Additionally, we also prompt the VLM to return the orientation of $o_{\text{in-hand}}$ during the manipulation phase, which would usually affect the success of tool use tasks.

Drawing inspirations from Manuelli et al. [32], we use the vector from x_{grasp} to x_{function} to specify the orientation of the object and ask the VLM to predict the orientation from a finite set of options (e.g. forward, backward, upside, downside, left, right).

Since robust grasping relies on contact physics and gripper design, which is beyond the capability of existing pre-trained VLMs, we combine the VLM predictions with analytical approaches in robotic grasping pipelines. Similar to Manuelli et al. [32], we use a grasp sampler to propose candidate grasps based on local geometric information from the observed point cloud. Instead of directly relying on the predicted x_{grasp}, we use the position and orientation of the grasp candidate that is closest to x_{grasp}. For more implementation details, please refer to Appendix B.

E. Bootstrapping through Physical Interactions

We consider two ways of further bootstrapping the performance of MOKA through physical interactions with the real world. In both ways, we unroll the VLM policy for the target task to collect robot experiences. The success labels of the collected experiences will be annotated by the VLM or humans.

In-context learning. We use a handful of successful trajectories as in-context examples to guide the high-level and low-level reasoning of the VLM. As discussed in prior work [4], two to three such in-context examples can significantly boost the performance of the VLM. Without changing the prompts p_{high} and p_{low}, we append the prompt with three pairs of annotated images and VLM responses from previous successful rollouts by the VLM policy.

Policy distillation. Given successful rollouts of the target task, we can train a student policy to imitate the VLM's successful trajectories. Effectively applying such an approach usually requires data of much larger scales, which often needs substantial efforts in data collection. This approach can showcase how MOKA can be used to collect and bootstrap on real-world robotic datasets for future imitation learning and reinforcement learning algorithms.

V. Experiments

The primary goal of our experiments is to validate and analyze the behavior of MOKA on open-vocabulary tasks with a wide range of objects and skills. We design our experiments to investigate the following questions:

- Can MOKA effectively perform affordance and motion reasoning on 2D images for open-vocabulary tasks?
- After translating the prediction of MOKA into low-level motion, how well does it perform on the task we proposed?
- Can MOKA improve from real-world trials, either via in-context learning or policy distillation?

We first evaluate MOKA's performance against baseline methods on 4 manipulation tasks in both zero-shot and in-context learning settings. Then we provide qualitative and robustness analysis in Sec. V-C.

Each of our evaluation tasks has two stages, and each task involves a variety of object interaction and tool-use scenarios. A summary of our evaluation tasks can be found in Tab. II. We also tested additional open-vocabulary tasks to demonstrate our method, which can be found in Appendix C.

A. Experimental Setup

We compare MOKA with **Code-as-Policies** [25] and **VoxPoser** [17], two baselines that also enable zero-shot execution of open-vocabulary tasks:. Code-as-Policies provides a framework for language model-generated programs executed on robotic systems by prompting with code examples. For a fair comparison, we provide the two baselines with the task description in the code comments, with an additional 40 lines of code prompts providing example usage, as in the original implementation [25]. Similarly, VoxPoser [17] also provides code examples to large language models to build a 3D voxel map of value functions. For VoxPoser, we reuse the example prompts and planning pipeline in the original implementation, and use the same hyper-parameters to create the voxel value map. For both baselines, we only adapt the perception modules for fair comparisons, while retaining the functionality of the other components.

We evaluate MOKA in both zero-shot and in-context learning settings, and refer to them as **MOKA zero-shot** and **MOKA in-context**, respectively. For the in-context learning

Methods	Table Wiping		Watch Cleaning		Gift Preparation		Laptop Packing	
	Subtask I	Subtask II	Subtask I	Subtask II	Subtask I	Subtask II	Subtask I	Subtask II
Code-as-Policies [25]	0.7	0.6	0.6	1.0	1.0	0.7	0.4	0.8
VoxPoser [17]	0.6	0	0.6	0.8	1.0	0.6	0.5	0.8
MOKA Zero-Shot (Ours)	0.6	0.6	0.7	1.0	1.0	0.7	0.5	0.8
MOKA Distilled (Ours)	**1.0**	0.7	0.8	0.8	1.0	0.7	1.0	**1.0**
MOKA In-Context (Ours)	0.9	**0.9**	**0.9**	**1.0**	**1.0**	**0.9**	**1.0**	0.9

TABLE I: Success rate of our method and baselines. Across 4 tasks, each consists of 2 subtasks, MOKA consistently achieves superior performances. We also demonstrated that the performance can be further bootstrapped through physical interactions in the environment using policy distillation and in-context learning.

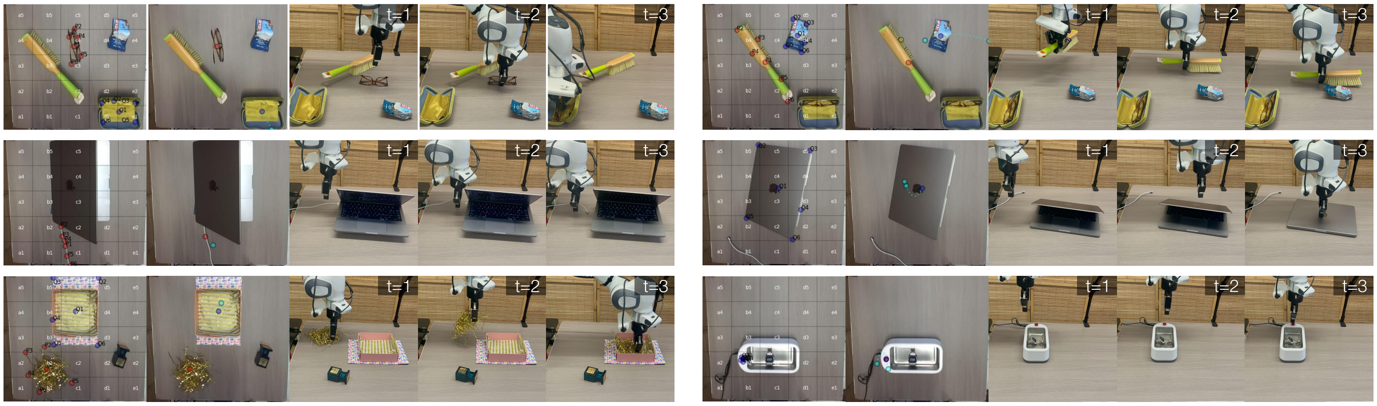


Fig. 3: Example results of the predicted keypoints and motions. On each row, two individual subtasks are displayed. For each subtask, we plot the marked image, the predicted point-based affordances, and three key frames in the trajectory.

Table Wiping	1. Move away the glasses
	2. Sweep the trash with the broom
Watch Cleaning	1. Put the watch into the ultrasound cleaner
	2. Press the power button
Gift Preparation	1. Put the golden filler in the gift box
	2. Put the perfume in the gift box
Laptop Packing	1. Unplug the cable
	2. Close the lid of the laptop

TABLE II: The subtask instructions of each of the testing tasks. Each of the testing tasks consists of two subtasks.

setting, we provide two examples to GPT-4V with instructions and annotated images before querying information about the current task. The annotated images are collected in scenes with object and instruction variations. Some in-context prompt examples can be found in Appendix.

The successful trajectories generated by MOKA can also serve as demonstration data for other learning-based methods. By simply training a model using the successful trajectories generated from MOKA, we can also "distill" the knowledge from a VLM to a learned policy. To achieve this goal, we employ a recent open-sourced robot foundation model Octo [35], which is pre-trained on a diverse mix of 800K robot trajectories [36] and can be easily fine-tuned to new tasks. Octo [35] supports language instructions or goal images as task specifications, and generates actions observations with a transformer-based diffusion policy architecture. It also provides code examples to easily perform fine-tuning. To construct our fine-tuning dataset, we collect 50 successful trajectories for each task with language annotations. We then adopt its latest base model checkpoint[1], and finetune the full model with recipe provided in the official instruction[2]. The performance of fine-tuned can be found in Sec. V-B with more implementation details in Appendix B-E. Although MOKA is a training-free method, it can be leveraged for data-driven approaches to provide training data of open-vocabulary tasks. We refer to this approach as MOKA-Distilled.

B. Quantitative Evaluation

Our quantitative evaluation results across 4 tasks are illustrated in Tab. I. For each task, we report the number of successes out of 10 trials. As shown in the the table, MOKA achieves state-of-the-art performance at each subtask of the 4 tasks (totally 8 subtasks), with consistent improvements using in-context learning. On most of the tasks, VoxPoser [17] has similar performance with MOKA (zero-shot), except for subtask 2 of table wiping (which is a tool-use task). Additionally, the task success rates can be sensitive to the resolution of the voxel map, which requires some hyperparameter tuning. Unlike the baselines, MOKA can work well without example prompts, or can achieve better performance with just two clean and intuitive example prompts.

[1] https://huggingface.co/rail-berkeley/octo-base
[2] https://github.com/octo-models/octo

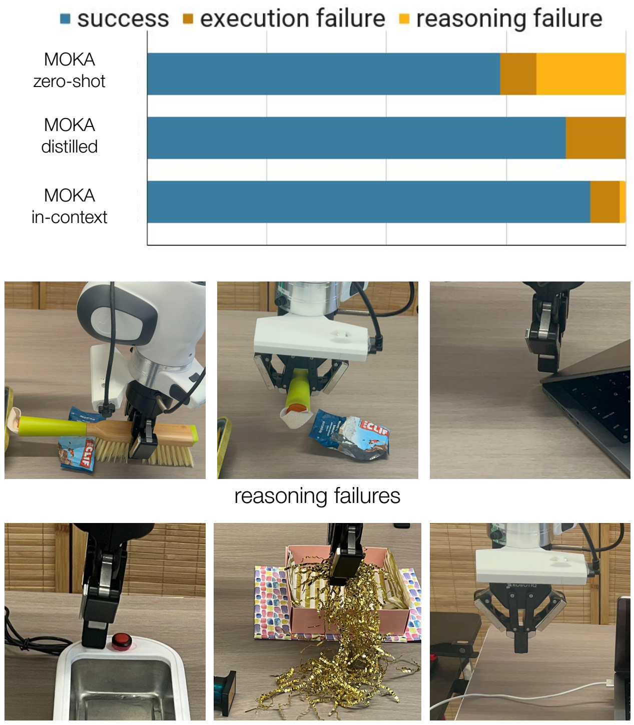


Fig. 4: **Failure analysis.** We breakdown the failure cases of the variants of our approach using zero-shot prediction, distilled policies, and in-context learning. We breakdown the failures into reasoning failures (caused by errors in the affordance prediction) and execution failures (caused by low-level motions).

We also observed that we can obtain an effective end-to-end policy by finetuning a pre-trained robot policy [35] using the successful trajectories generated by MOKA. This suggests that the generated data is of high quality, and thus can be used as demonstration data for open-vocabulary tasks. The fact that distilling the successful MOKA trials actually *improves* the overall performance of the system suggests the feasibilty of using MOKA as a "data generator", where the zero-shot MOKA method is used to bootstrap a continuous improvement system. This is an exciting direction to explore in future work.

Failure breakdown We analyze the failure cases of MOKA. Trajectories with wrong predictions from the GPT-4V are counted as reasoning failures. The following failure cases are illustrated in the top row of Fig. 4, including grasping the broom upside down due to confusing the the grasp point with the function point, pointing the room to the wrong direction due to wrong target angle, pressing the hinge of the laptop due to the wrong target point. The trajectories with desired VLM prediction but fail to execute successfully are counted as execution failures. The following failure cases are illustrated in the bottom row of Fig. 4. From left to right, the failures include the gripper narrowly misses the button, the filler getting disassembled in the middle of the trajectory, and the cable slipped through the gripper fingers.

As shown in the figure, both policy distillation and in-context learning reduces the total failures. Using the distilled policy, the VLM is no longer part of the pipeline. Therefore there is no reasonoing failures in this case. But the execution failures increases since the policy does not realy on the open-vocabulary detection and segmentation.

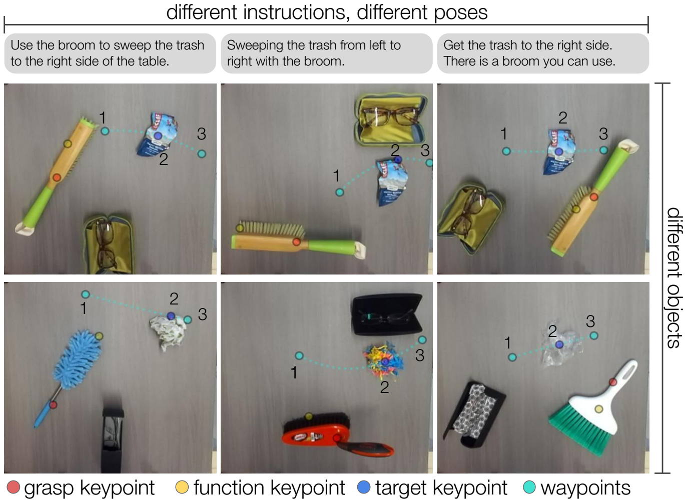

different instructions, different poses
Use the broom to sweep the trash to the right side of the table.
Sweeping the trash from left to right with the broom.
Get the trash to the right side. There is a broom you can use.
different objects
1
2
3
● grasp keypoint ● function keypoint ● target keypoint ● waypoints

Fig. 5: **Robustness analysis.** We analyze MOKA's robustness with respect to various instructions, initial arrangements, and objects. Each column in the image uses the same language instruction and similar initial arrangements of objects. The two rows involve different objects.

C. *Qualitative Evaluation*

In Fig. 3, we provide six examples of the annotated marks, generated motions, and the unrolled trajectories of MOKA. In each of these examples, based on the annotated marks, MOKA successfully selected the keypoints and waypoints for generating the desired motions. These examples demonstrate that our approach can be applied to various skills including pressing, closing, rearranging, tool use, etc.

We analyze the robustness of MOKA with respect to instruction variations, initialization variations, and object variations. Fig. 5 provides a pictorial illustration of MOKA's predictions under different instructions and observations. Each image is queried with the language prompt on the top within the same column. The examples in the same column share similar initialization object positions and orientations. The example in the first row uses the same set of objects while the second row involves objects of alternative geometries, colors, and materials. Some of these objects are also deformable or transparent. Fig. 5 demonstrates consistent point predictions within rows and columns, indicating that MOKA is robust to the changes of the language instructions, initialization, and objects for the same task.

VI. Conclusion and Discussion

In this paper, we proposed MOKA, a simple and effective visual prompting method that leverages VLMs for robot manipulation. By representing manipulation tasks with point-based affordances, we convert the motion generation process to a visual question-answering problem that VLMs can solve. MOKA provides a general and flexible framework that can intuitively and effectively harness VLM to generate point-based motion for a wide range of open-vocabulary tasks, while preserving the visual reasoning capabilities of VLMs. Our experiments demonstrate the effectiveness and robustness of MOKA across multiple tasks in both zero-shot and in-context learning manners. As far as we know, MOKA is the very first method that leverages visual prompting on VLMs for open-vocabulary robot manipulation.

The diversity and difficulty of solvable tasks for MOKA depend on the capabilities of VLMs. For example, we mostly evaluate MOKA in 4-DoF table-top manipulation tasks, as current VLMs are not capable of reliably predicting 6-DoF motions. It will also be hard to extend MOKA to more challenging scenarios, e.g. bi-manual manipulation tasks, where the coordination of two arms requires more complex 3D understanding. In addition, the dependency of querying third-party VLM APIs also constraints us from applying MOKA to dynamic tasks, with low latency tolerance. We believe the above limitations can be conquered with more capable and accessible VLMs in the future.

In our experiments, we also demonstrate that the trajectories generated by VLM can be further used to train a learning-based policy that yields better performance. We hope MOKA can inspire future research toward designing more efficient and scalable algorithms to collect and bootstrap on real-world open-vocabulary datasets more autonomously.

VII. Acknowledgement

Fangchen Liu is supported in part by NSF under the NSF AI4OPT Center. We would like to thank Karl Pertsch, Oleg Rybkin, and Zheyuan Hu for providing feedback on early drafts of the manuscript. We also want to thank Sudeep Dasari, Qiyang Li, Yide Shentu, and Homer Walke for valuable suggestions on the method and task design.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[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.

[3] Aitor Aldoma, Federico Tombari, and Markus Vincze. Supervised learning of hidden and non-hidden 0-order affordances and detection in real scenes. In *2012 IEEE international conference on robotics and automation*, pages 1732–1739. IEEE, 2012.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S Ryoo, Austin Stone, and Daniel Kappler. Open-vocabulary queryable scene representations for real world planning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11509–11522. IEEE, 2023.

[6] Changhyun Choi and Henrik I Christensen. Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation. In *2010 IEEE International Conference on Robotics and Automation*, pages 4048–4055. IEEE, 2010.

[7] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24 (240):1–113, 2023.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[10] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23171–23181, 2023.

[11] James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology press, 2014.

[12] Tucker Hermans, James M Rehg, and Aaron Bobick. Affordance prediction via learned object attributes. In *IEEE international conference on robotics and automation (ICRA): Workshop on semantic perception, mapping, and exploration*, pages 181–184. Citeseer, 2011.

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

[14] Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. *arXiv preprint arXiv:2311.17842*, 2023.

[15] 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.

[16] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson,

Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[17] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[18] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[19] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[20] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

[21] Feng Li, Hao Zhang, Peize Sun, Xueyan Zou, Shilong Liu, Jianwei Yang, Chunyuan Li, Lei Zhang, and Jianfeng Gao. Semantic-sam: Segment and recognize anything at any granularity. *arXiv preprint arXiv:2307.04767*, 2023.

[22] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022.

[23] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[24] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975, 2022.

[25] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy 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.

[26] Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2motion: From natural language instructions to feasible plans. *arXiv preprint arXiv:2303.12153*, 2023.

[27] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[28] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[29] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[30] Parsa Mahmoudieh, Deepak Pathak, and Trevor Darrell. Zero-shot reward specification via grounded natural language. In *International Conference on Machine Learning*, pages 14743–14752. PMLR, 2022.

[31] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *2010 IEEE International Conference on Robotics and Automation*, pages 2308–2315. IEEE, 2010.

[32] Lucas Manuelli, Wei Gao, Peter R. Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *International Symposium of Robotics Research*, 2019. URL https://api.semanticscholar.org/CorpusID:80628296.

[33] Stephen Miller, Mario Fritz, Trevor Darrell, and Pieter Abbeel. Parametrized shape models for clothing. In *2011 IEEE International Conference on Robotics and Automation*, pages 4861–4868. IEEE, 2011.

[34] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection. In *European Conference on Computer Vision*, pages 728–755. Springer, 2022.

[35] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. https://octo-models.github.io, 2023.

[36] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[37] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[38] Zengyi Qin, Kuan Fang, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Keto: Learning keypoint representations for tool manipulation, 2019.

[39] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[40] Alec Radford, Jeffrey Wu, Rewon Child, David Luan,

Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9, 2019.

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[42] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[43] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.

[44] Yujun Shi, Chuhui Xue, Jiachun Pan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. *arXiv preprint arXiv:2306.14435*, 2023.

[45] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh 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.

[46] Austin Stone, Ted Xiao, Yao Lu, Keerthana Gopalakrishnan, Kuang-Huei Lee, Quan Vuong, Paul Wohlhart, Brianna Zitkovich, Fei Xia, Chelsea Finn, et al. Open-world object manipulation using pre-trained vision-language models. *arXiv preprint arXiv:2303.00905*, 2023.

[47] Jie Sun, Joshua L Moore, Aaron Bobick, and James M Rehg. Learning visual object categories for robot affordance prediction. *The International Journal of Robotics Research*, 29(2-3):174–197, 2010.

[48] Jur Van Den Berg, Stephen Miller, Ken Goldberg, and Pieter Abbeel. Gravity-based robotic cloth folding. In *Algorithmic Foundations of Robotics IX*, pages 409–424. Springer, 2010.

[49] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[50] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[51] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023.

[52] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chun yue Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *ArXiv*, abs/2310.11441, 2023. URL https://api.semanticscholar.org/CorpusID:266149987.

[53] Lingfeng Yang, Yueze Wang, Xiang Li, Xinlong Wang, and Jian Yang. Fine-grained visual prompting. *arXiv preprint arXiv:2306.04356*, 2023.

[54] Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 9(1):1, 2023.

[55] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[56] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

[57] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

[58] Xueyan Zou, Jianwei Yang, Hao Zhang, Feng Li, Linjie Li, Jianfeng Gao, and Yong Jae Lee. Segment everything everywhere all at once. *arXiv preprint arXiv:2304.06718*, 2023.

Appendix A
Experiment Details

A. Environment

Our experiments are conducted in a real-world table-top manipulation environment, which involves a 7-DoF Franka Emika robot arm with a 2F-85 Robotiq gripper interacting with various objects on the table at 5Hz. Our tabletop environment has two fixed ZED 2.0 cameras and one ZED mini wrist camera that can take RGBD images.

The *top-down camera*, which is the primary camera used in this paper, provides RGB and the depth images for MOKA and other baseline methods.

In addition, we also set up the front camera and the wrist camera for more complete observation to distill the collected robot experiences to the learned Octo policy [35]. The action space of the robot is 7-dimensional, consisting of the 6DoF end-effector twist defined in the Cartesian space with an additional dimension of gripper position. Each task can consist of multiple stages, each terminates within a fix horizon of 100 steps.

B. Task Design

We design various table-top manipulation tasks with a diverse set of daily objects. Tab. III shows the language description of the full list of tasks.

Table Wiping	1. Move the glasses into the glasses case
	2. Sweep the trash with the broom
Watch Cleaning	1. Put the watch into the ultrasound cleaner
	2. Press the power button
Gift Preparation	1. Put the golden filler in the gift box
	2. Put the perfume in the gift box
Laptop Packing	1. Unplug the cable
	2. Close the lid of the laptop
Fur Removing	1. Grasp the fur remover
	2. Sweep the fur on the sweater with the remover
Drawer Closing	1. Close the drawer
Scissor Handing	1. Grasp the scissor
	2. Hand the scissor to a human
Flower Arrangement	1. Grasp the pink roses on the table
	2. Insert the roses into the vase

TABLE III: The language description of all the proposed tasks. Each of the tasks consists of two stages except for the drawer closing task.

We provide the comparative evaluation on the top four tasks (Table Wiping, Laptop Packing, Gift Preparation and Untrasound Cleaning) in the main paper, with additional results of MOKA on other 4 tasks in our supplementary video.

C. Success Checking

To count the failure modes and collect successful trajectories, we first query VLM with task instruction and obtain the point-based motion prediction. If the prediction is correct, it will be executed on the robot. Otherwise, it will be counted as the VLM reasoning failure as mentioned in Sec. V-B. After executing the motion prediction from VLM, the human expert will manually check if the task succeeds or not. If the task is not successfully finished, it will be counted as an execution failure. All the successful trajectories will be saved as demonstrations for policy distillation.

Appendix B
Implementation Details

We introduce the overview of MOKA in Alg. 1, and the implementation details of each component in the following sections.

Algorithm 1 MOKA Pipeline

```
Input: Vision-language Model $\mathcal{M}$, Task instruction $l$, text prompt for high-level reasoning $p_{high}$, text prompt for low-level reasoning $p_{low}$, initial observation $s_0$
Query $\mathcal{M}$ for high-level task reasoning, obtain $y_{high} = \mathcal{M}([p_{high}, l, s_0])$ which decompose the task into $N$ subtasks.
for subtask $k = 0 \cdots N-1$ do
    Get observation $s_k$ from the top-down camera
    Propose keypoint and waypoint candidates and get annotated image $f(s_k)$
    Query $\mathcal{M}$ for low-level motion reasoning, obtain $y^k_{low} = \mathcal{M}([p_{low}, y^k_{high}, f(s_k)])$
    Execute $y^k_{low}$ on the real robot
end for
```

A. High-level Task Reasoning

Given the initial observation image s_0 and the language task description l, we first query the VLM $\mathcal{M}$ with the language instruction to produce the decomposed subtask, which contains the structured information for the K subtasks, including descriptions of objects and desired motion. We provide the high-level reasoning we used across all the tasks in Tab. IV.

The response contains fields "object_grasped", "object_unattached" and "motion_direction", which will be used for later stages in our pipeline, such as keypoint proposal and motion prediction.

B. Point-based Affordance Proposal

After obtaining the high-level reasoning results, we can know the objects involved in each subtask. Since VLM cannot directly generate keypoints and waypoints, we need to propose some candidate points and let VLM select corresponding points through a visual question-answering way.

Keypoint proposal. We leverage GroundedSAM [43] to extract segmentation masks conditioned on a text prompt, which is designed to be a string of object names involved in the current subtask (e.g. "trash, broom"). Given an image of current observation and such a text prompt about the involved object names, we first employ GroundingDINO [27] to generate bounding boxes for the objects by conditioning on the text prompt. Later, the annotated boxes given by GroundingDINO [27] serve as the box prompts for SAM [18] to generate corresponding segmentation masks for the objects.

The input request contains:

- A string describes the multi-stage task.
- An image of the current table-top environment captured from a top-down camera.
- List of object candidates in the scene.

The output response is a list of dictionaries in the JSON form. Each dictionary specifies the information of a subtask, following the correct order of executing the subtasks to solve the input task. Each dictionary contains these fields:

- **instruction**: A string to describe the subtask in natural language forms.
- **object_grasped**: A string to describe the name of the object that the robot gripper will hold in hand while executing the task (e.g., the object to be picked or used as a tool to interact with other objects). This field can be empty if there is no such object involved in this subtask.
- **object_unattached**: A string to describe the name of the object that the robot gripper will interact with directly or via another object without holding it in hand (e.g., the object to be touched by the tool, the target object where "object_grasped" will be moved onto). This field can be empty if there is no such object involved in this subtask.
- **motion_direction**: A string to describe the direction of the robot gripper motion while performing the task (e.g., "from right to left", "backward", "downward").

TABLE IV: The high-level reasoning prompt for all the tasks. It will decompose a multi-stage task into subtasks. The specified output fields can be used for later low-level reasoning stage.

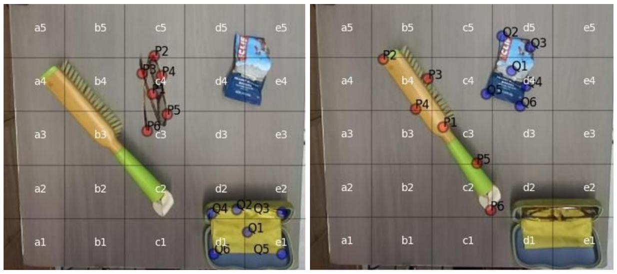


Fig. 6: The affordance proposal for the table wiping task in stage 1 and stage 2. The keypoints are plotted on corresponding objects for different stages based on the high-level reasoning results. The annotated grid cells are used for VLM to select the position of potential waypoints.

We define the keypoints of each object to be a set of boundary keypoints with a center keypoint. To extract them, we sample K points on the contour of each object using farthest point sampling [37], and also include the geometric mean point of the object segmentation mask. We then plot these $K+1$ keypoints on the 2D image as our keypoint proposals to the VLM.

Waypoint proposal. Unlike keypoints, waypoints are often the points in the free space that are not attached to any objects. To perform waypoint selection, we evenly divide the full image into 5×5 grid tiles, which mark the column as *a, b, c, d, e* from left to right and rows as *1, 2, 3, 4, 5* from bottom to top, as shown in Fig. 6. The waypoints will be sampled from the predicted tile from VLM.

C. Low-level Motion Reasoning

After annotating the observation image using object keypoints and grid cells as described in the above section, we can query VLM about low-level motion with annotated image and text prompts. Firstly, we describe the text and image inputs to the VLM using prompt V.

Then we first explain the definition of keypoints and waypoints using prompt VI.

After that, we can specify the output format and further explain the role of each field using prompt VII. We can also provide some step-by-step guidance about how the reasoning procedure should be done (similar to chain-of-thought prompting [50]). Our complete low-level reasoning prompts are covered in Tab. V, VI and VII.

D. Motion Execution

We can decompose a manipulation subtask into an **optional** grasping phase and a manipulation phase. For some tasks like tool using, the robot needs to first grasp an object before making contact with another object. For other tasks where the robot can directly interact with target objects (e.g. pushing, pressing button), the "grasp_keypoint" field will be empty. For such tasks, we will skip the execution of grasping phase.

Grasping phase. We first sample 30 antipodal 4DoF grasp proposals based on the primary camera's depth image, which is cropped by the bounding box of **object_grasped**. Here, we use the same antipodal grasp proposal method used in DexNet 2.0 [29]. We then lift the VLM-predicted grasp point from 2D image to the robot frame based on the primary camera's intrinsic and extrinsic parameters. After that, we can select the nearest grasp proposal based on the lifted grasp point, and execute the 4DoF grasp on the robot.

Manipulation phase. After obtaining the pre-contact tile and post-contact tile, we first sample pre-contact waypoint and post-contact waypoint from the predicted tiles. Since the waypoints are in the free space, its 3D position cannot be determined given its 2D projection on the image. So we also assign the VLM-predicted height to them. We only consider the cases where the waypoints are in same height with target point (e.g. pushing), or above the target point (e.g. placing) in most common table top manipulation scenarios.

We then sequentially move the function keypoint to the pre-contact waypoint, target keypoint and post-contact waypoint to perform a contact motion.

After both phases are successfully done, the robot need to get the motion prediction for the next subtask. In order to obtain a clean and non-occuluded image to perform low-level reasoning, we first move the robot to the neural pose, get an

Please describe the robot gripper's motion to solve the task by selecting keypoints and waypoints.
The input request contains:

- The task information with these fields:
  - **instruction**: The task in natural language forms.
  - **object_grasped**: The object that the robot gripper will hold in hand while executing the task.
  - **object_unattached**: The object that the robot gripper will interact with either directly or via another object without holding it in hand.
  - **motion_direction**: The motion direction of the robot gripper or the in-hand object while performing the task.
- An image of the current table-top environment captured from a top-down camera, annotated with a set of visual marks:
  - **candidate keypoints on object_grasped**: Red dots marked as $P[i]$ on the image.
  - **candidate keypoints on object_unattached**: Blue dots marked as $Q[i]$ on the image.
  - **grid for waypoints**: Grid lines that uniformly divide the images into tiles. The grid equally divides the image into columns marked as a, b, c, d, e from left to right and rows marked as 1, 2, 3, 4, 5 from bottom to top.

TABLE V: The description about image and text inputs in the low-level motion reasoning stage.

The motion consists of an optional grasping phase and a manipulation phase, specified by **grasp_keypoint**, **function_keypoint**, **target_keypoint**, **pre_contact_waypoint**, and **post_contact_waypoint**.
The definitions of these points are:

- **grasp_keypoint**: The point on "object_grasped" indicates the part where the robot gripper should hold.
- **function_keypoint**: The point on "object_grasped" indicates the part that will make contact with "object_unattached".
- **target_keypoint**: If the task is pick-and-place, this is the location where "object_grasped" will be moved to. Otherwise, this is the point on "object_unattached" indicating the part that will be contacted by "function_keypoint".
- **pre_contact_waypoint**: The waypoint in the free space that the functional point moves to before making contact with the "target_keypoint".
- **post_contact_waypoint**: The waypoint in the free space that the functional point moves to after making contact with the "target_keypoint".

TABLE VI: The explanation about the defination of keypoints and waypoints.

The response should be a dictionary in JSON form, which contains:

- **grasp_keypoint**: Selected from candidate keypoints marked as $P[i]$ on the image. This will be empty if and only if object_grasped is empty.
- **function_keypoint**: Selected from candidate keypoints marked as $P[i]$ on the image. This will be empty if and only if object_grasped or object_unattached is empty.
- **target_keypoint**: Selected from keypoint candidates marked as $Q[i]$ on the image. This will be empty if and only if object_unattached is empty.
- **pre_contact_tile**: The tile that the pre-contact waypoint should be in. This is selected from candidate tiles marked on the image.
- **post_contact_tile**: The tile that post-contact waypoint should be in. This is selected from candidate tiles marked on the image.
- **pre_contact_height**: The height of pre-contact waypoint as one of the two options "same" or "above" (same or higher than the height of making contact with target keypoint).
- **post_contact_height**: The height of post-contact waypoints as one of the two options "same" or "above".
- **target_angle**: Describe how the object should be oriented during this motion in terms of the axis pointing from the grasping point to the function point.

Think about this problem step by step and explain the reasoning steps. First, choose grasp_keypoint, function_keypoint, and target_keypoint on the correct parts of the objects. Next, describe which tile the target_keypoint is located in. Then choose pre_contact_tile, post_contact_tile, pre_contact_height, post_contact_height such that the resultant motion from pre-contact waypoint to target keypoint, then to post-contact waypoint in 3D follows the "motion_direction" input. Remember that the columns are marked as 'a', 'b', 'c', 'd', 'e' from left to right, and the rows are marked as 1, 2, 3, 4, 5 from bottom to top.

TABLE VII: The output format of the low-level motion reasoning, including some further explanations.

observation image from the top-down camera, and then resume the robot. Subsequently, the robot will execute the predicted motion until the multi-stage task is finished.

E. Policy Distillation

We employ the base checkpoint of Octo [35] and perform full-model finetuning using their official instruction at https://github.com/octo-models/octo. The default architecture of Octo is a transformer-based diffusion policy, which is conditioned on a task discription or goal image through a task tokenizer along with tokenized pixel observations from multiple cameras, and predicts actions through a diffusion head. The diffusion head consists of a 3-layer MLP with a hidden dimension of 256 using a standard DDPM objective [13].

We mostly reuse the hyperparameters as in Octo [35]. For customized hyper-parameters we used, please refer to Tab. VIII.

Hyperparameter	Value
Learning Rate	3e-4
Warmup Steps	1000
Weight Decay	0.01
Gradient Clip Threshold	0.5
Batch Size	256
Total Gradient Steps	200000

TABLE VIII: Hyperparameters used during fine-tuning.

F. *Evaluation*

We evaluate MOKA in both zero-shot and in-context learning settings. For the zero-shot setting, we keep all the above prompts **unchanged** but only change the language task description (e.g. "sweep the garbage", "pick up the perfume", "press the button").

For in-context learning settings, we first collect two examples from VLM's successful predictions in different scenes with scene and object variations. As shown in Fig. 7, MOKA can be improved from such simple and intuitive in-context examples, without intricate prompt engineering.

APPENDIX C
ADDITIONAL RESULTS

A. *Ablation Study*

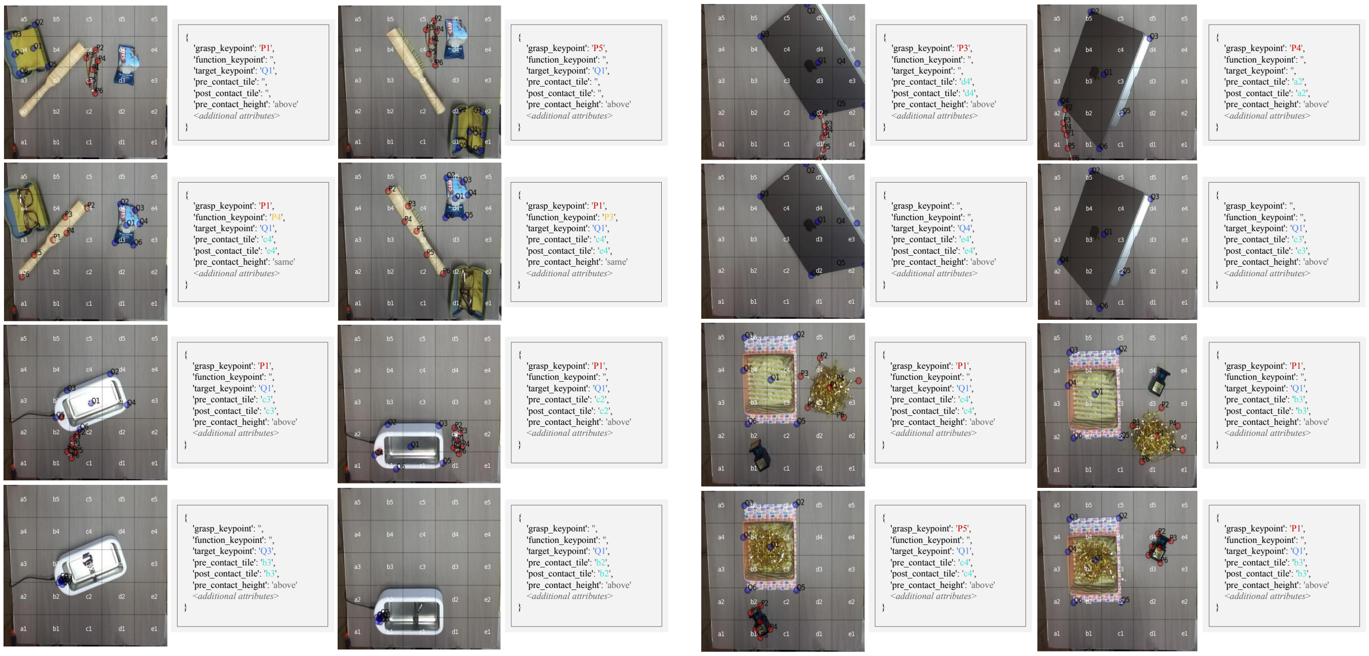
We design the following ablative studies to understand the effectiveness of different design options in MOKA.

- MOKA w/o hierarchy: we can skip the high-level task reasoning but directly ask for low-level motion reasoning from GPT-4V. Here all the objects in the scene will be annotated with keypoints, and VLM is queried to generate point-based affordance directly from the annotated image.
- MOKA w/o description of points: we can remove the description of the definition of keypoints and waypoints in Tab. VI.
- MOKA w/o chain-of-thought prompting: we can remove the step-by-step guidance at the last paragraph in the prompt in Tab. VII.

The results of our ablative studies are illustrated in Tab. IX. For each task, we report the number of reasoning successes rate out of 10 trials. The results demonstrate that our method can obtain consistent improvements from all the above prompting designs. Specifically, **MOKA w/o hierarchy** decreases the performance by a large margin, which indicates the importance of subtask decomposition. **MOKA w/o keypoint description** and **MOKA w/o CoT** can preserve most of the performance, but still worse than MOKA on most of the tasks. As illustrated in Fig. 8, VLM can make various mistakes in terms of keypoint and waypoint predictions. Specifically, for MOKA w/o hierarchy, the VLM can make mistakes about subtask ordering, which can cause a complete failure of the task.

B. *Additional Tasks*

For our four additional tasks, please refer to our supplementary video.



```
{
  'grasp_keypoint': 'P1',
  'function_keypoint': '',
  'target_keypoint': 'Q1',
  'pre_contact_tile': '',
  'post_contact_tile': '',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P5',
  'function_keypoint': '',
  'target_keypoint': 'Q1',
  'pre_contact_tile': '',
  'post_contact_tile': '',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P3',
  'function_keypoint': '',
  'target_keypoint': '',
  'pre_contact_tile': 'd4',
  'post_contact_tile': 'd4',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P4',
  'function_keypoint': '',
  'target_keypoint': '',
  'pre_contact_tile': 'a2',
  'post_contact_tile': 'a2',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P1',
  'function_keypoint': 'P4',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'c4',
  'post_contact_tile': 'e4',
  'pre_contact_height': 'same'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P1',
  'function_keypoint': 'P3',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'c4',
  'post_contact_tile': 'e4',
  'pre_contact_height': 'same'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': '',
  'function_keypoint': '',
  'target_keypoint': 'Q4',
  'pre_contact_tile': 'e4',
  'post_contact_tile': 'e4',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': '',
  'function_keypoint': '',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'c3',
  'post_contact_tile': 'e3',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P1',
  'function_keypoint': '',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'c3',
  'post_contact_tile': 'c3',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P1',
  'function_keypoint': '',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'c2',
  'post_contact_tile': 'c2',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P1',
  'function_keypoint': '',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'c4',
  'post_contact_tile': 'c4',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P1',
  'function_keypoint': '',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'b3',
  'post_contact_tile': 'b3',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': '',
  'function_keypoint': '',
  'target_keypoint': 'Q3',
  'pre_contact_tile': 'b3',
  'post_contact_tile': 'b3',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': '',
  'function_keypoint': '',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'b2',
  'post_contact_tile': 'b2',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

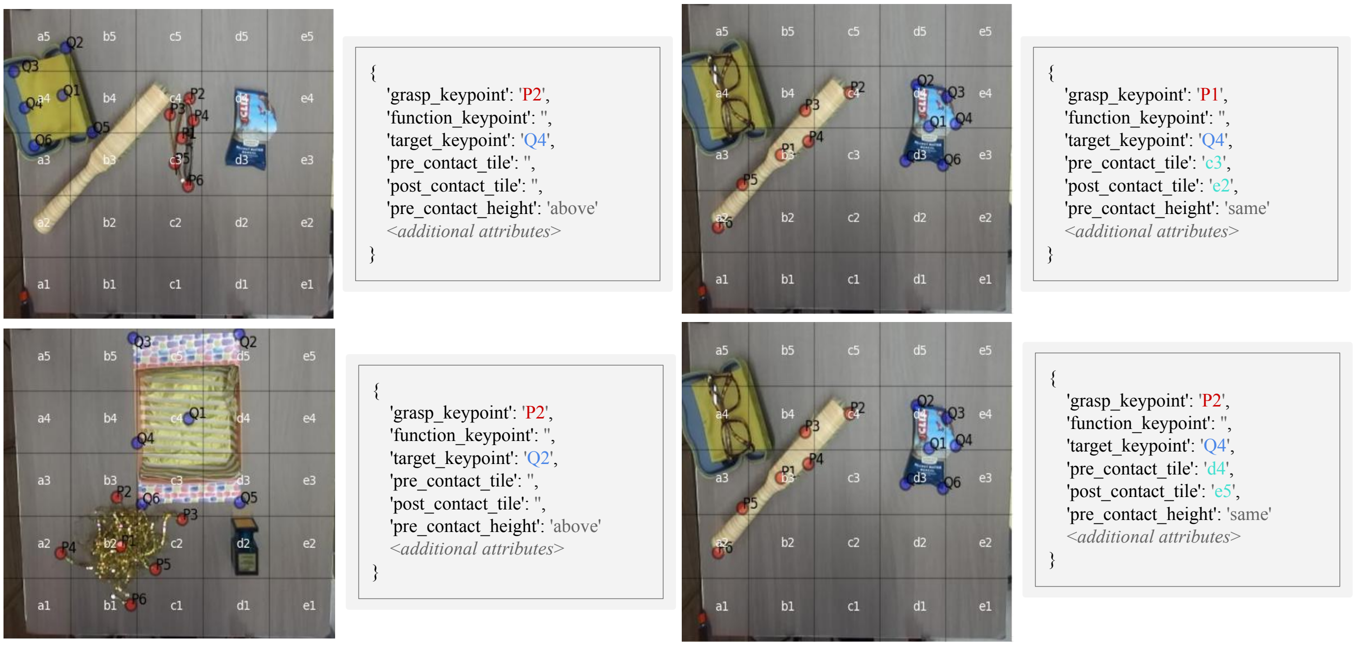
```
{
  'grasp_keypoint': 'P5',
  'function_keypoint': '',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'c4',
  'post_contact_tile': 'c4',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P1',
  'function_keypoint': '',
  'target_keypoint': 'Q1',
  'pre_contact_tile': 'b3',
  'post_contact_tile': 'b3',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

Fig. 7: The in-context examples used in MOKA which are collected under object pose variations.

	Table Wiping		Watch Cleaning		Gift Preparation		Laptop Packing	
	Subtask I	Subtask II	Subtask I	Subtask II	Subtask I	Subtask II	Subtask I	Subtask II
MOKA	0.7	0.7	0.8	1.0	1.0	0.8	0.6	0.8
MOKA w/o hierarchy	0.1	0.1	0.2	0.1	0.2	0.2	0.0	0.1
MOKA w/o keypoint description	0.5	0.6	0.7	0.8	0.9	0.6	0.4	0.8
MOKA w/o CoT	0.5	0.4	0.6	0.8	0.9	0.6	0.4	0.6

TABLE IX: Ablation studies on different prompt designs. Across 4 tasks, each consists of 2 subtasks, MOKA consistently benefits from the three prompt designs.



```
{
  'grasp_keypoint': 'P2',
  'function_keypoint': '',
  'target_keypoint': 'Q4',
  'pre_contact_tile': '',
  'post_contact_tile': '',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P1',
  'function_keypoint': '',
  'target_keypoint': 'Q4',
  'pre_contact_tile': 'c3',
  'post_contact_tile': 'e2',
  'pre_contact_height': 'same'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P2',
  'function_keypoint': '',
  'target_keypoint': 'Q2',
  'pre_contact_tile': '',
  'post_contact_tile': '',
  'pre_contact_height': 'above'
  <additional attributes>
}
```

```
{
  'grasp_keypoint': 'P2',
  'function_keypoint': '',
  'target_keypoint': 'Q4',
  'pre_contact_tile': 'd4',
  'post_contact_tile': 'e5',
  'pre_contact_height': 'same'
  <additional attributes>
}
```

Fig. 8: Qualitative results of ablation studies. Without keypoint description or chain-of-thought prompting, the model can make mistakes in keypoint prediction (left column) and waypoint prediction (right column).