VLMbench: A Benchmark for Vision-and-Language Manipulation

Kaizhi Zheng  
University of California, Santa Cruz  
kzheng31@ucsc.edu

Odest Chadwicke Jenkins  
University of Michigan, Ann Arbor  
ocj@umich.edu

Xiaotong Chen  
University of Michigan, Ann Arbor  
cxt@umich.edu

Xin Wang  
University of California, Santa Cruz  
xwang366@ucsc.edu

1. Introduction

Recent progress in embodied AI [2, 5–7] pushes intelligent robotic systems to reality closer than any other time before. To achieve the final goal of interacting with unstructured environments to accomplish various daily tasks, the agent needs to learn how to manipulate objects through visual observations and natural languages appropriately. Compared with vision-only systems, natural language possesses two essential properties that enhance robot manipulation task learning: compact and flexible specification of various tasks and natural interactive communication interface with humans. We name such an agent that learns from the combined knowledge of language and vision to accomplish robot manipulation tasks a Vision-and-Language Manipulation (VLM) embodied agent. Given the rapid growth in VLM research and their limitations on generalization across different tasks, we expect an inclusive, modular, and scalable benchmark to evaluate embodied agents for various manipulation tasks.

Compared to recent benchmarks developed to evaluate robotics manipulation tasks with language guidance and visual input [1, 3, 8], which lack (1) adaptation to novel objects automatically and (2) categorization for modular and flexible composition to complex tasks, we present VLMbench, a highly categorical robotic manipulation benchmark that generates numerous task demonstrations, as shown in Fig. 1. To generalize across the different object and task settings, we propose AMSolver, an automatic unit task builder that can generate various robot trajectories and grasping poses to accomplish the desired action for a unit task. Compared with previous works [3] that require the user to design a new task from scratch, we propose unit task templates that can generate a new task simply by specifying object properties. For instance, the pick-place task can be automatically generated for different objects provided their geometric models, with free variation in the objects’ colors, shapes, and sizes and 6 degrees of freedom (DoF) poses in the scene. The unit tasks can be composed together to create complex multi-step tasks. Besides, VLMbench provides both high-level language descriptions of the entire task and low-level action descriptions that correspond to unit robot moves.

2. AMSolver: Automatic Manipulation Solver

We consider three main categories of household manipulation tasks: rearrangement (move an object from place to place, e.g., collect trash into the bin), affordance usage (exploit particular usage of some object, e.g., pour water from a mug), and multi-step tasks (can be decomposed into several individual steps, e.g., connect every two pieces to assemble a table from parts). To formulate a unified task representation, we assume that every task within these categories can...
be considered a combination of objects and unit actions. To describe the essential elements of tasks, we proposed Automatic Manipulation Solver (AMSolver), a rule-based automatic task builder consisting of object-centric representations and unit task templates, shown in Fig. 2.

**Rule-based Unit Tasks** We define a *unit task* as the semantic step of completing a sub-goal of the entire task. Specifically, a unit task is defined in a formula of ‘take action on an object under certain constraints’ where a unit task can be parameterized by two constraints: (a) **position constraints** and (b) **orientation constraints**, which describe the valid spatial space or orientation range, respectively, of the end-effector for a specific task. We propose three unit task templates detailed below that can compose the aforementioned complex tasks: **Control** is a preparation or ending step of a task, which models the transition of the object state, where the state indicates whether the robot can move the object or not; **M1** denotes moving the target object with goal pose constraints (position and orientation constraints), which can be modeled as a 6 DoF transform in the robot’s workspace; **M2** denotes moving the target object along a trajectory while satisfying the motion constraints during the entire path, which implies a more strict condition than **M1**.

**Object-centric Representation** Some recent works [4, 9] have used object-centric representations for manipulation. Since the properties are defined on objects, these representations can easily cross the variations of environments, agents, and tasks. Our benchmark assumes that objects used in the tasks are rigid, and their fundamental properties will not change during the tasks. Therefore, we can parameterize the objects as a set of configurations, including class, color, size, and geometry shape. If the object is articulated, its whole configuration will contain the configuration of each part and the physical constraint of each connection.

**3. VLMbench: Visual-and-Language Manipulation Benchmark**

Given language instructions, the Vision-and-Language Manipulation (VLM) task requires an embodied agent to follow the instructions to complete tabletop manipulation tasks. Formally, at the beginning of the task, the agent receives a set of language instructions \( L = \{L_1, L_2, ..., L_n\} \), where \( L_i \) denotes one sentence of arbitrary length. The initial state \( s_0 \) contains multi-view RGB images, depth images, segmentation information, and robot states, including joint angles, velocities and torques, and end-effector pose. Given the observations and language instructions, the agent needs to estimate an executable action command \( a_0 \), directly working on the end-effector or joints. Then, at each step \( t \), the agent receive new observations \( o_t \) and generate the action \( a_t = f(s_t, s_0, s_1, ..., s_{t-1}, L) \) for the next step. The step loop will repeat until the agent sends a stop action or should be terminated, e.g., achieve the success condi-

![Figure 2. The unit task templates in the AMSolver. On the left, we show three main common task types of household tasks and give several example tasks for each. On the right, we show three unit task templates parameterized by position and orientation constraints over the robot end-effector on either goal pose or entire path. By combining these unit task templates, various task examples can be generated.](image-url)
References


