2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

LLaKey: Follow My Basic Action Instructions to Your Next Key State

Zheyi Zhao[1], Ying He[1], Fei Yu[1†], Pengteng Li[1], Fan Zhuo[1], and Xilong Sun[1]

***Abstract*— In 3D object manipulation, collecting expert data for end-to-end imitation learning becomes a mainstream method. Though successful, previous works neglect the guiding role of language in action execution. These methods lack the understanding of action semantics, in which multiple action sequences are guided by a category of instructions, resulting in overlearned object semantics and vague action semantics. To address the above limitation, we introduce a novel framework named LLaKey, which breaks down skill commands into more detailed action instructions based on key states for fine-grained action control. Specifically, LLaKey first leverages the knowledge encoded in pre-trained large-scale models to fine-tune an action instruction conductor. Then, these instructions are executed by a downstream action model. Comprehensive experiments show that LLaKey significantly surpasses baselines with a relative improvement of 15% in nine complex and varied skill tasks, demonstrating the superiority of our method.**

I. Introduction

In the field of 3D object manipulation, collecting expert data for end-to-end imitation learning is a standard approach. This strategy allows for the direct emulation of expert techniques, essential for developing more sophisticated manipulation models. A significant portion of current research is devoted to two main areas: enhancing the representation of 3D information, as indicated by studies like [1], and semantically enriching databases, as in the case of [2]. These efforts aim to improve the accuracy and complexity of interactions between robotic systems and their 3D environments.

Though successful, previous works neglect the guiding role of language in action execution leading to performance degradation. These methods lack the understanding of action semantics, where multiple action sequences are guided by a category of instructions, resulting in overlearned object semantics and vague action semantics. As shown in Figure 1, we investigate the existence of this problem by directing two different action sequences with the same type of skill instructions: 'put the {color} light bulb in the {color} box', which are filled based on the operated object and the target location. One of the sequences includes an unscrewing action, while the other consists solely of pick-and-place actions. Without altering the skill instructions of the test set, the success rate is 60%. Subsequently, we maintain the model constant but alter the language instructions of the test set to '{color} light bulb, {color} box', keeping only the operative object and the target location, which increases the success rate to 64%. It suggests that the model focuses more on object semantics. After retraining with two types of language instructions: 'put the {color} light bulb in the {color} box' and 'unscrew the {color} light bulb and put it into the {color} box', the success rate remains at 60%. However, when we replace or remove the verbs in the language input, such as 'take the {color} light bulb and it into the {color} box', for the test set, the success rate unexpectedly increases to 72% instead of decreasing. It confirms that the model excessively focuses on the semantics of objects in the language and has a vague understanding of action semantics. Lacking the understanding of action semantics makes the model lack the ability to modify the sequence and permutation of actions to adapt to variations in the object's environment. Therefore, the model training process requires stronger language guidance. Each skill during its execution will have multiple key states, such as a pre-grasping state where the velocity of all joints decreases to zero. Each key state can be achieved by the execution of one or two basic actions. We define the following as basic actions: moving (significant pose changes with minor rotational adjustments), rotating (major rotational changes with minimal pose alterations), grasping (pose movement towards the gripper accompanied by gripper closure), and releasing (pose movement towards the gripper along with gripper opening). We believe that skill instructions should

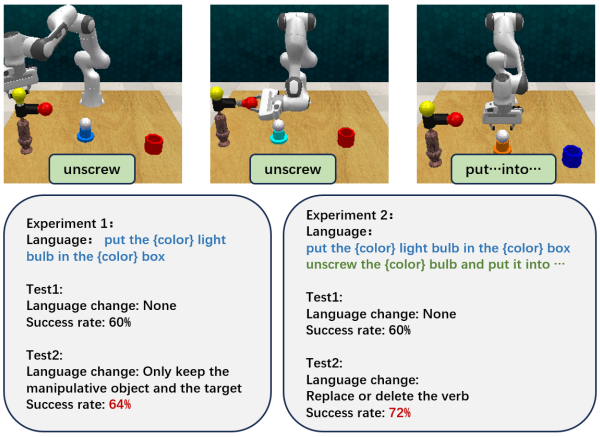

Fig. 1: **The overview of our empirical exploration for exploring insufficient action semantic learning.** We train two skill action sequences using both single-type and dual-type languages. During the testing process, we alter language instructions to demonstrate the existence of the problem. The results indicate that model's understanding of action semantics is ambiguous.

[1]Z. Zhao, Y. He, P. Li, F. Zhuo, and X. Sun are with the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen University, Shenzhen, Guangdong 518060, P.R. China; zhaozheyi@gml.ac.cn ; heying@szu.edu.cn; 2110276192@email.szu.edu.cn; zhuofan@gml.ac.cn;sunxilong1988@163.com

[2]F. Yu is with the College of Computer Science and Software Engineering, Shenzhen University, P.R. China, and also with Carleton University, Canada yufei@szu.edu.cn

[2]F. Yu† is the corresponding author.

be further broken down into simpler, more basic action commands for improved learning and application. Each key state should have its own language goal, then each action can correspond to its true semantic meaning and effectively utilize the action knowledge learned previously.

To overcome these challenges mentioned above, we propose a novel framework named LLaKey, which gains a deep understanding of basic action semantics to obtain higher intelligence. LLaKey is composed of an upstream command model and a downstream action model. The command model is a fine-tuned LLM that receives skill instructions and outputs the next basic action instruction based on the current environment. The instructions from the command model are then passed to the action model for execution. This approach can be applied to large models that lack inherent coding capabilities, fully leveraging their extensive language knowledge. The action model has a deep understanding of action semantics, which each action is designed to match its true semantic meaning in its training proces. As both the command and action models possess a strong understanding of actions, LLaKey can modify the sequence and permutation of actions to adapt to variations in the object's environment, thereby progressively advancing the completion of the skill.

To summarize, our contributions are threefold:

- We introduce an innovative framework named ***LLaKey***. This marks the first proposal of the "basic actions" concept in a model, fostering the model's learning of higher-level and more flexible skill semantics.
- To solve the insufficient action semantic learning, we decompose skill instructions into basic action instructions to form a new dataset. It successfully fine-tunes the command capabilities of the large model and enhances the action model's understanding of action semantics.
- Extensive experiments show that our LLakey can outputform the baseline, showing promising results in both simulations and real-world applications.

The rest of this paper is organized as follows: Section II presents the related work. Section III discusses the methodology. Section IV details the experiments conducted. Section V concludes the paper and discusses its limitations.

II. Related Work

A. 3D Object Manipulation

End-to-end image-to-action policy models, such as RT-1 [3], GATO [4], BC-Z [5], and InstructRL [6], directly predict 6-DoF end-effector poses from 2D video and language inputs. They require many thousands of demonstrations to learn spatial reasoning and generalize to new scene arrangements and environments. Expanding on RT-1, MOO [7] uses OWL-ViT [8] to provide (x, y) image coordinates for objects mentioned in the instructions. This spatial information is added as an additional channel to the image data. C2F-ARM [9] and PerAct [10] voxelize the point clouds and use a 3D convolutional network as the backbone for control inference. RVT [11] addresses the issue of memory consumption by transforming the point cloud into a set of RGB-D images from multiple views. Act3D [1] employs CLIP pretrained 2D backbones to process 2D image views and uses depth information [12] to lift these 2D features into 3D.

B. LLM Based Agents

With the impressive emergent capabilities and increasing popularity of large language models (LLMs) [13], [14], [15], researchers begin to integrate these models into the development of artificial intelligence agents [16], [17], [18]. They employ LLMs as central elements in the cognitive frameworks or as controllers for these agents. Such LLM-integrated agents exhibit reasoning and planning skills akin to those of symbolic agents, which employs methods such as Chain-of-Thought (CoT) and problem decomposition [19], [20]. LLM agents are leveraging GPT-4 [15]'s coding capabilities in various innovative ways. For instance, CaP [21] and ProgPrompt [22] recursively generates code using provided APIs, instruct2act [23] and Socratic model [24] employs the SAM [25] and CLIP models, and Voxposer [26] uses LLMs to form voxel value maps for robotic navigation and task execution. Language Models as Zero-Shot Planners [27] mimic writing based on new tasks by leveraging language inference. SayCan [28] combines task progression scoring with a value function for success prediction, while Tidybot [29] infers personal preferences for object placement. Successdetectors [30] enhances the precision of robotic tasks by fine-tuning the Flamingo [31] model for success detection. RT2 [32] is a visual language model that outputs low-level robot actions and addresses large-scale tasks, working in tandem with PaLM-E [33] for refining multimodal sentences, thereby enhancing the interaction between visual inputs and language model outputs for precise and context-relevant robotic actions.

III. Method

Our complete method is illustrated in Figure 2. LLaKey comprises two parts: the upstream command model and the downstream action model. For the upstream command model, the inputs include images from various perspectives as well as the current task, and its goal is to generate the next basic action instruction. The downstream action model then takes this action instruction, along with point cloud data, to predict a movement specified by a target end-effector pose and gripper state.

A. Skill Command Decomposition

We define that the expert data for a task can be represented as a video stream. This video is decomposed into several streams based on keyframes, where each key frame signifies crucial or bottleneck steps in the gripper's task execution. These steps include actions like pre-pick, grasp, place pose, or instances where all joint velocities are zero. As shown in Figure 3, the basic action instructions are descriptions of the fundamental actions that occur to reach the target state.

Given that action instructions serve as a medium for information transfer between the upstream and downstream models, each verb does not require many variants, reducing

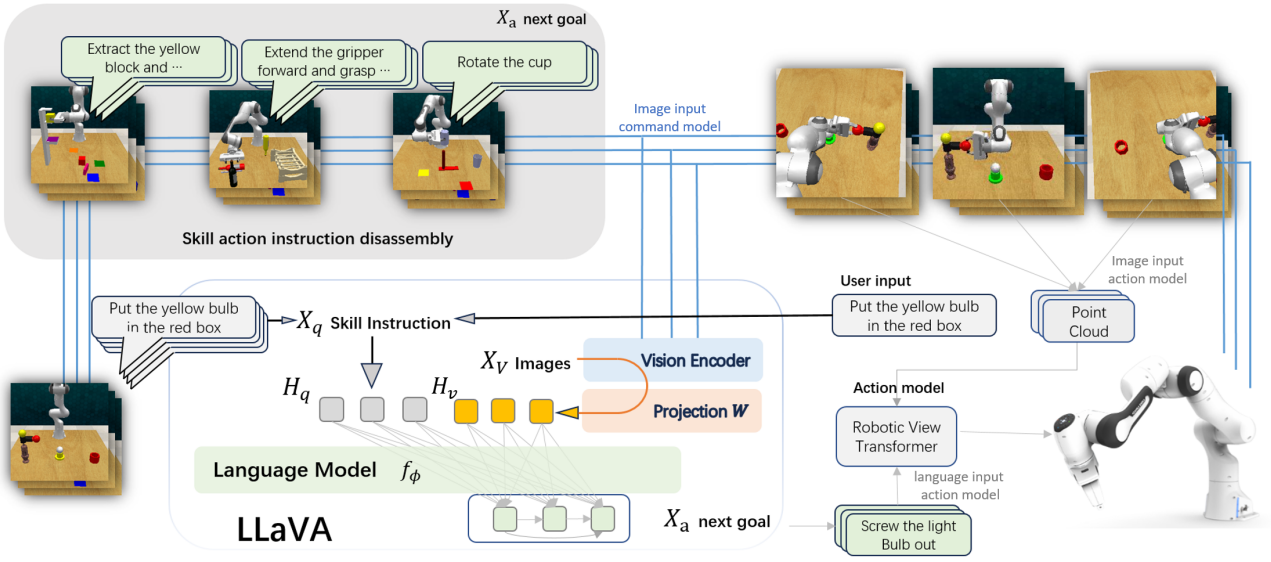

Fig. 2: **Overview of our proposed LLaKey.** In the training pipeline, expert data is decomposed at keyframes and annotated to form image-text pairs. These pairs are used for fine-tuning multimodal large models as the upstream command model. The downstream action model is then trained using the decomposed action instructions as new linguistic information. In the testing pipeline, the upstream command model outputs the next action instruction, which is executed by the downstream action model.

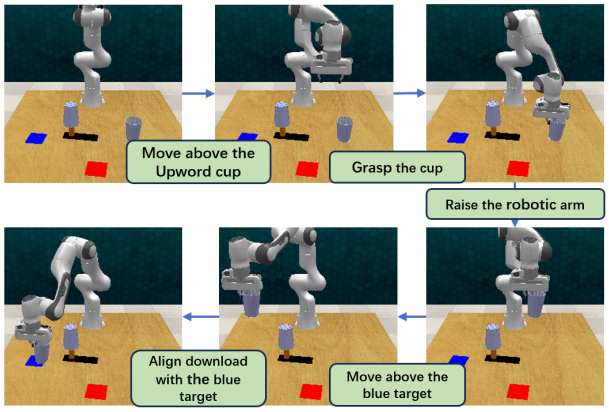

Fig. 3: **An example of skill decomposition for the command "move the upward cup to blue target".** Ignoring the starting point, this skill consists of five key states. The content within the green boxes represents the corresponding basic action instructions that have been decomposed.

the cognitive load for both models. If the number and sequence of basic actions in the decomposed video streams are identical, then their corresponding basic action instructions must also be similar.

B. Training Dataset Construction

We have at our disposal a dataset of skill instructions $\mathcal{D} = \{(\xi_1, g_1), (\xi_2, g_2), \ldots, (\xi_n, g_n)\}$, comprising n instances of expert demonstrations, each correlated with English language goals $\mathcal{G} = \{g_1, g_2, \ldots, g_n\}$. These demonstrations have been curated by an expert leveraging a motion planner to attain specific intermediate poses. Each demonstration ξ consists of a series of continuous actions $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ alongside corresponding observations $\mathcal{O} = \{o_1, o_2, \ldots, o_i\}$. An individual action a includes parameters such as the 6-DoF pose, state of the gripper (open or closed), and the use of collision avoidance by the motion planner to achieve an intermediate pose, denoted by $a = \{a_{\text{pose}}, a_{\text{open}}, a_{\text{collide}}\}$. Observations o entail RGB-D imagery from an array of cameras, utilizing four cameras in simulated experiments $o_{\text{sim}} = \{o_{\text{front}}, o_{\text{left}}, o_{\text{right}}, o_{\text{wrist}}\}$ and two cameras in real-world scenarios $o_{\text{real}} = \{o_{\text{front}}, o_{\text{workspace}}\}$.

In every demonstration ξ, we identify a subset of pivotal actions $\{k_1, k_2, \ldots, k_m\} \subset \mathcal{A}$ as keyframes. Each keyframe action k is linked to a specific basic action instruction I, which explicates the primary movements executed from the preceding state to the current one, noted as $\{I_1, I_2, \ldots, I_m\} \subset \mathcal{I}$.

At intervals of every ten frames within each demonstration ξ, we extract the corresponding observation o to construct a dataset consisting of $\{o, k_{\text{next}}, I_{\text{next}}, g\}$. Here, k_{next} represents the action required to proceed to the next key state, I_{next} represents the textual description of this action, and g represents the skill task instruction.

C. Fine-Tuning Multimodal Large Models

The upstream command model of LLaKey takes images from the mechanical arm's left and right shoulders, as well as images facing the direction of the arm. These images, combined with instructions, are fed into the projection module, forming the large model's input tokens. The output is basic action instructions, which are then taken as linguistic inputs by the downstream action model.

We use LLaVA [34] as our pre-trained multimodal large model in our research. LLaVA integrates LLaMa2-chat [35] as our "brain", and the pre-trained CLIP visual encoder ViT-L/14 as our perception component.

Specifically, for an input image X_v, it is encoded into a visual feature $Z_v = g(X_v)$. The grid features before

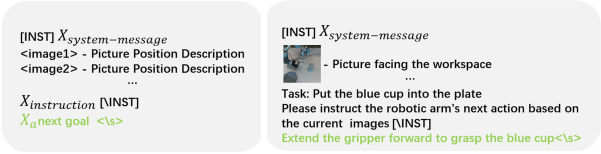

Fig. 4: **The input sequence is used to fine-tune the model.**

and after the last Transformer layer are connected into the word embedding space of the language model through a simple linear layer, which is a trainable projection matrix W. This converts Z_v into language embedding tokens H_q that match the dimensionality of the word embedding space in the language model:

$$H_v = W \cdot Z_v, \quad \text{with} \quad Z_v = g(X_v) \tag{1}$$

To define a target for Vision-Language Model (VLM) fine-tuning, we construct our dataset as shown in Figure 4 to fine-tune LLaVA. This includes system messages, images from multiple perspectives of the robotic arm, and corresponding descriptions of the image locations. In our current implementation, $X_{\text{system-message}}$ is set to "You are a helpful language and vision assistant. You are able to understand the visual content that the user provides and assist the user with a variety of tasks using natural language." The stop token is defined as STOP = $<\backslash$s$>$. $X_{\text{instruction}}$ is formulated as "Task: {user input}, please instruct the robotic arm's next action based on the current images." The model is trained to predict the next action description and identify where to stop. Consequently, only the green sequence/tokens are utilized to compute the loss in the auto-regressive model.

It is noteworthy that, to equip LLaVA with the capability for action commanding, two distinct abilities are essential. The first is the error correction ability, which is the capacity to judge whether an action command has been completed. The second, termed sequential prediction ability, is crucial for accurately predicting the next action command upon the completion of a current one. The variation in distances between keyframes across different tasks, especially when some are extremely close to each other, leads to data scarcity, presenting challenges for LLaVA. For obtaining the mentioned above abilities, we first ensure that the amount of data between keyframes is sufficient to address data imbalance, essential for the error correction ability. Then, for the sequential prediction ability, we replicate the data from keyframe to keyframe multiple times, thereby increasing the sampling rate and maintaining a balance between error correction and action prediction.

D. Action

We choose RVT as our action model. Notably, in the training data of RVT, the timestep of the current environment in the task is also included. However, since LLaKey focuses on performing actions based on action instructions, the action model does not need to know which phase of a complex task it is in, so we opt to remove this timestep information. For the loss function, we employ the cross-entropy function on the heat maps of each image. The ground truth is modeled using a truncated Gaussian distribution around the 2D projection of the 3D ground-truth location, with the variance gradually decreasing during training until it stabilizes at a fixed value.

Algorithm 1 LLaKey

Require:
$token(-)$: Text Encoder
$g(-)$: Visual Encoder
W: Projection matrix for visual features
$f(-)$: Large Language Model (LLM)
$Action(-)$: Action Prediction Model
Input:
C: A user command
$X_v^{1\ldots m}$: Images from m different perspectives
$D_v^{1\ldots m}$: Descriptions for $X_v^{1\ldots m}$
Parameters:
I_{next}: Next action instruction
A: 6-DoF pose, state of the gripper

$I_{pre} = 0$
while not end of task **do**
  prompt $= \bigoplus_{i=1}^{m}(\text{token}(D_v^i) + W \cdot g(X_v^i)) + token(C)$
  $I_{\text{next}} = f(\text{prompt})$
  $A = Action(I_{\text{next}}, X_v^{1\ldots m})$
  if specific conditions in I_{next} **then**
    Modify A accordingly
  end if
  if $I_{\text{next}} = I_{pre}$ **then**
    number$+ = 1$
  end if
  if number $>$ threshold **then**
    $A = A + A_{\text{random}}$
  end if
  $I_{pre} = I_{\text{next}}$
  Execute A in the environment, updating state
end while

The action model evaluates whether basic action instructions include non-quantitative language, such as "grab" and "release", before it executes the action. If such language is present, the model adjusts the related joints accordingly. It's important to note that if command model does not recognize the completion of the current instruction, it may repeatedly send the same one. If the action model's response to the current environment and instruction is minimal, the entire framework could stall at this point. To address this issue, we enable the action model to track the frequency of each instruction. If an action instruction is repeated more than a specified threshold, the robotic arm's gripper is programmed to probabilistically move to an area within a certain distance from the gripper's end. This strategy is designed to introduce new visual features for command model and the action model, thereby enhancing their effectiveness.

TABLE I: **Multi-Skill Performance and Comprehensive Predictive Capability**. In the notation 'Model (N)', 'N' denotes the number of images received by the command model.

Task Type	Model (N)	Move Block	Move Cup	Put Bulb	Stack Wine	Close Jar	Open Drawer	Put in Cupboard	Screw Bulb	Sort Shape	Average
Multi-Skill	LLaKey13B (3)	**96**	**91**	**98**	**98**	**78**	**79**	33	**59**	51	**75.9**
	RVT[1]	84	52	52	84	42	72	**64**	42	**52**	60.4
Comprehensive Predictive Capability	LLaVA7B (3)	90.1	**87.0**	86.6	**86.1**	88.1	88.6	70.2	**86.0**	**78.2**	84.5
		88.7	90.5	88.6	86.1	78.1	**90.1**	74.2	82.0	**78.0**	84.0
	LLaVA13B (3)	**94.6**	86.8	**88.4**	84.7	**91.1**	**89.8**	**71.6**	85.3	77.3	**85.5**
		87.6	**92.6**	**92.0**	**86.7**	**78.6**	87.8	**76.6**	**88.3**	77.8	**85.3**
	LLaVA13B (1)	67.4	57.8	66.8	62.0	66.6	76.1	32.9	64.6	34.7	58.7
		79.0	64.1	81.1	69.4	61.0	83.0	33.0	53.1	25.0	60.9
	GPT4-V (3)	3.8	0	0	0	0	22.1	0	3.2	2.3	3.5
		3.2	0	0	0	0	21.0	0	3.0	2.5	3.3

1 denotes using point clouds as input.

IV. Experiments

A. Simulation Experiments

Environment. The simulation is set in CoppelaSim [36] and interfaced through PyRep [37]. All experiments use a Franka Panda robot with a parallel gripper. The input observations are captured from four RGB-D cameras positioned at the front, left shoulder, right shoulder, and on wrist. All cameras are noiseless and have a resolution of 128 × 128.

Language-Conditioned Tasks. We train and evaluate 5 RLBench [38] tasks, including picking and placing, bulb screwing, and drawer opening. Each task includes several variations, ranging from 2-60 possibilities. In the 5 Rlbench tasks, different variations involve selecting different target positions or different objects to manipulate, but the action sequence remains constant. All keyframes from an episode have the same language goal, which is constructed from templates (but human-annotated for real-world tasks). These five tasks each have a fixed sequence of basic actions, hence are single-sequence tasks. Additionally, we create four multi-action sequence tasks.

Figure 5(a) has 3 action sequences: inserting a red block into a purple target, sliding a red block into an orange target, and pulling a yellow block out toward a blue or green target. All these sequences correspond to the skill instruction "move the color block to color target". Figure 5(b) has 2 action sequences where red and yellow light bulbs need to be unscrewed, all corresponding to the skill instruction "put the color light bulb to the color target". Figure 5(c) has 4 action sequences involving inverting cups needing to be pulled out and rotated, and cups facing upwards needing to be clamped or slid, corresponding to the skill instruction "move the pose cup to the color target". Figure 5(d) has 3 action sequences, corresponding to the skill instruction "stack the color wine bottle to the target of the rack". In total, these custom multi-action sequence tasks comprise 4 types of instructions, 12 action sequences, and 35 variations.

These variants are randomly sampled during data generation but kept consistent during evaluations for one-to-one comparisons. Note that in all experiments, we do not test for generalization to unseen objects, our train and test objects are the same. However, during test time, the agent has to handle novel object poses, randomly sampled goals, and randomly sampled scenes with different semantic instantiations of object colors, shapes, sizes, and categories.

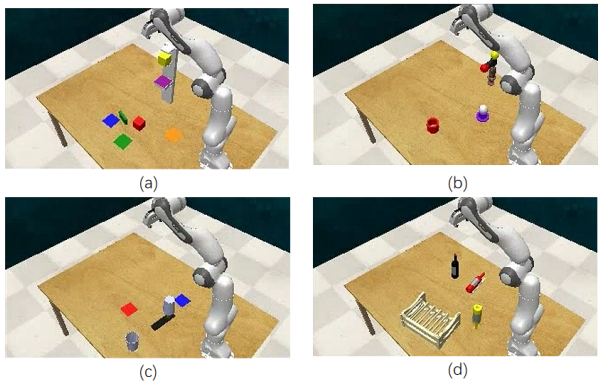

Fig. 5: **Overview of our proposed customized multi-action sequence operational tasks.** (a) Move block to a specific color target, where the yellow block needs to be pulled out. (b) Move light bulb to a box, with red and yellow light bulbs needing to be unscrewed. (c) Move cup to a specific color target, where the upside-down cup needs to be rotated. (d) Place the wine bottle on a specific spot on the rack, with the yellow wine bottle requiring rotation.

B. Training and Testing

Baseline In an evaluation of performance on 18 tasks in RLBench, the success rates of various models are as follows: Image-BC achieves a success rate of only 1.3%, C2F-ARM-BC reaches 20.1%, PerAct attains 49.4%, and RVT outperforms them with a success rate of 62.9%. Therefore, for our comparison, we focus solely on RVT.

Training and Evaluation Details. Just like the baseline, we use the RLBench training dataset with 100 expert demonstrations per task (900 demonstrations over all tasks). We train the RVT model for 100k steps with a batch size of 20, employing new basic instructions while keeping the rest of the training configuration unchanged.

For LLaVA, 200 expert data samples are provided for custom tasks, and 100 expert data samples for the rest of the tasks, with each expert data sample taken every 10 frames. There must be no fewer than 3 data samples between each keyframe, and any shortfall is randomly supplemented. The data from each keyframe to the next is repeated 6 times. We only keep the visual encoder weights frozen and continue to update both the pre-trained weights of the projection layer and LLM in LLaVA. For action instructions, we provide 2 to 3 semantically similar variants for each verb. In total, we design 16 templates, including single actions like "move to the {pose} of {something}", "rotate {something} to {something}", "grasp (or release) {something}", and combinations of these actions. We also include actions with implicit conditions, such as "slide {something} to {something}", "screw {something} in (or out)", "align downward with {something}", and specific robotic arm operations like "raise the robotic arm". Each template has at most one semantically equivalent sentence, tailored to meet the training requirements of large multimodal models.

For the assessment of multi-task performance, each test episode is strictly categorized as either a success or a failure. Our methodology involves computing average success rates over 25 distinct test episodes for each task, amounting to a total of 225 episodes across all tasks. To enhance the robustness of our evaluation, each of the 25 test episodes per task is repeated six times. We then discard the highest and the lowest scores from these repetitions to compute a more representative average. In this testing regime, LLaKey is programmed to continue executing actions until either task completion is confirmed by an oracle or a maximum of 25 steps is reached.

For testing the Sequential Prediction Ability and Error Correction Ability of LLaVA, we employ the same data collection methods for the test set as used in the training set. Specifically, for assessing the Sequential Prediction Ability, we test an average of 241 frames per task, corresponding to the white rows in Table I. In the case of the Error Correction Ability, we test an average of 512 frames per task, which corresponds to the gray rows in Table I. In addition, we also investigate the impact of different numbers of image inputs and varying model capacities of multimodal large models on these two abilities.

Result. Table I in the multi-skill section demonstrates the multi-skill performance of LLaKey, revealing that our method surpasses the baseline by 15.5%. Additionally, the "Comprehensive Predictive Capability" section shows the impact of varying numbers of input images and different sizes of multimodal large models on both abilities, where data on a white background represents sequential prediction ability and data on a gray background indicates error correction ability. We observe that a 13B model with inputs from three images taken from different perspectives achieves an effectiveness of 85% in both abilities. Figure 6 illustrates LLaKey's ability to roll back tasks in the event of unexpected situations, indicating its capability to re-arrange basic actions and re-execute skill tasks effectively in four types of unforeseen circumstances. Additionally, we train for direct action prediction with LLaVA, following the RT-2 approach. However, even after increasing the expert data for each task to 400, we do not achieve favorable results, thus, these results are not published in Table I.

C. Real-World Experiments

We assess the performance of LLaKey on real visual sensory data by training and testing the model in a real-world setup.

Real World Setup. We conduct our experiments on a tabletop using a statically mounted Universal Robots UR3 arm. The scene is perceived through a Realsense d435i (RGB-D) camera, statically mounted in a third-person view. Before feeding into the motion module for training, we calibrate the robot camera extrinsic and transform the perceived point clouds to the robot base frame.

Tasks. We design two skill tasks for manipulating toys and cups, respectively, with each task having two different operational sequences corresponding to the same type of skill command. As shown in Figure 7, (a) is one operation sequence of the toy manipulation task, where toy located inside a cabinet need to be extracted. Another involves the pick-and-place of toy on a table. Both operations correspond to the same skill command, "Move something into the plate" (b) is an operation sequence in the cup manipulation task, where an upside-down cup needs to be lifted, rotated, and then placed into the plate. Another operation involves a cup facing upwards, which is gripped from one side of the rim for pick-and-place. Both of these operations correspond to the skill command, "Put something into the plate".

Data Collection. We initially collect a dataset for the motion module training through human demonstration. Given a sequence of basic actions and scene configuration, we ask the human demonstrator to specify a sequence of gripper target poses by kinesthetically moving the robot arm. Once we have the sequence of target poses, we reset the robot to its start pose and then control it to sequentially move to each target pose in the specified order. We concurrently record the RGB-D stream from the camera as the robot moves toward the targets. For the commanding data of the multimodal large model, we instruct the robotic arm to move randomly within a 5cm radius horizontally at the position corresponding to each keyframe. We record images from two perspectives: facing the robotic arm and facing the workspace. These images are then manually annotated with basic action instructions. This provides us with a dataset of RGB-D frames paired with target pose and basic action instructions. In total, we collect 10 expert data for each action sequence.

Result. We apply the same training method to real-world data as we do in the simulation environment. For each operation sequence of every task, we conduct ten repeated tests. In the toy manipulation task, the action sequence involving the extraction of toys achieves a success rate of 100%, while the 2D pick-and-place operation on the table has a success rate of

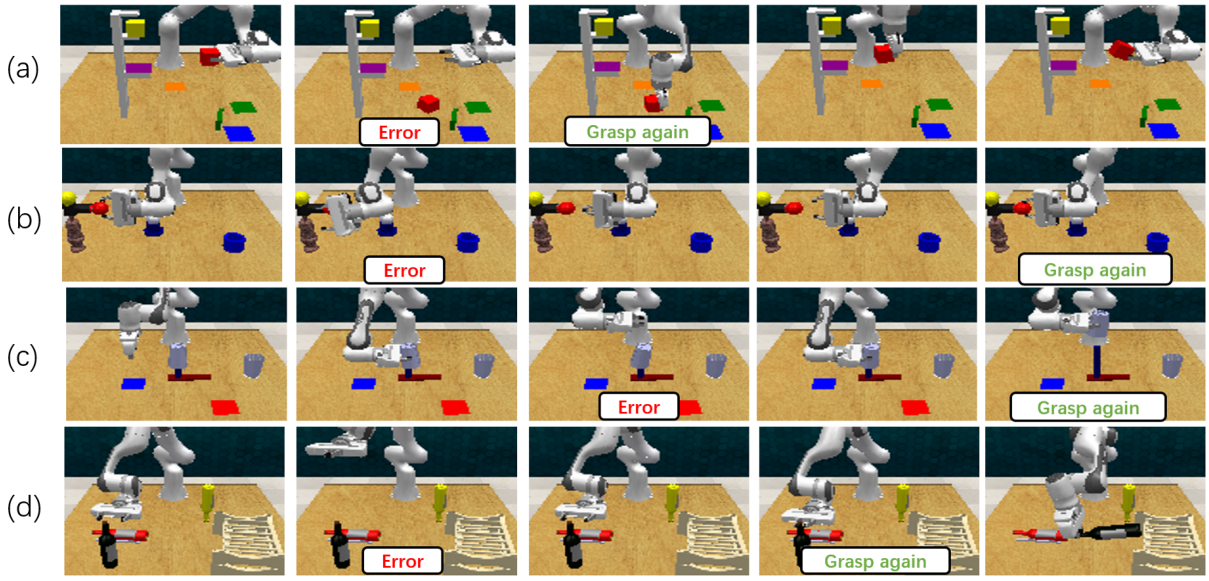

Fig. 6: **Overview of LLaKey's task rollback capability in response to unexpected situations.** (a) the object accidentally falls during movement, (b) a failed grasp that also obstructs the object, (c) a failed grasp that changes the object's position, and (d) a failed grasp with the gripper closing.

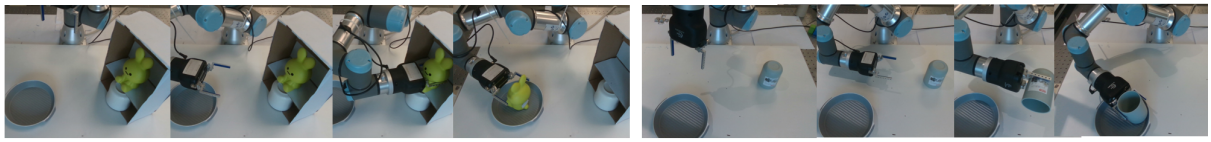

Fig. 7: **Examples of LLaKey in the real world.** (a) Extract toys from the cupboard, (b) Flip an upside-down cup.

60%. In the cup manipulation task, the short sequence pick-and-place task reaches a success rate of 80%, whereas the longer sequence involving lifting and inverting tasks achieves a success rate of 60%.

V. Conclusions and Limitations

We propose an innovative framework named LLaKey, which not only enhances the downstream action model's understanding of action semantics, thereby improving its task execution capabilities, but also stimulates the upstream commanding model's error correction ability and sequential prediction ability. It progressively advances the task towards the next key state until completion. The framework demonstrates outstanding capabilities in both simulated environments and the real world. Regarding limitations, our framework is constrained by the capabilities of the multimodal large model and the action model. We show poorer performance in tasks involving the stacking of multiple identical objects, as we cannot annotate the relationship between the positions of the objects and their stacking locations.

VI. Acknowledgment

This work is supported in part by Shenzhen Science and Technology Program under Grant ZDSYS20220527171400002, the National Natural Science Foundation of China (NSFC) under Grants 62271324, 62231020 and 62371309, and the Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515011979, the Hetao Shenzhen-HongKong Science and Technology Innovation Cooperation Zone(HZQB-KCZYZ-2021055), and Shenzhen Deeproute.ai Co., Ltd (HZQB-KCZYZ-2021055).

References

[1] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: Infinite resolution action detection transformer for robotic manipulation. *arXiv preprint arXiv:2306.17817*, 2023.

[2] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. *arXiv preprint arXiv:2309.01918*, 2023.

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

[4] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[5] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Proceedings of the Conference on Robot Learning*, pages 991–1002, 2022.

[6] Pierre-Louis Guhur, Shizhe Chen, Ricardo Garcia Pinel, Makarand Tapaswi, Ivan Laptev, and Cordelia Schmid. Instruction-driven history-aware policies for robotic manipulations. In *Proceedings of the Conference on Robot Learning*, pages 175–187, 2023.

[7] 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.

[8] M Minderer, A Gritsenko, A Stone, M Neumann, D Weissenborn, A Dosovitskiy, A Mahendran, A Arnab, M Dehghani, Z Shen, et al. Simple open-vocabulary object detection with vision transformers. arxiv 2022. *arXiv preprint arXiv:2205.06230*.

[9] Stephen James, Kentaro Wada, Tristan Laidlow, and Andrew J Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13739–13748, 2022.

[10] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the Conference on Robot Learning*, pages 785–799, 2023.

[11] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic view transformer for 3d object manipulation. *arXiv preprint arXiv:2306.14896*, 2023.

[12] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems*, 35:33330–33342, 2022.

[13] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[14] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[15] 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.

[16] Ruibo Liu, Ruixin Yang, Chenyan Jia, Ge Zhang, Denny Zhou, Andrew M Dai, Diyi Yang, and Soroush Vosoughi. Training socially aligned language models in simulated human society. *arXiv preprint arXiv:2305.16960*, 2023.

[17] Lilian Weng. LLM-powered Autonomous Agents. *lilianweng.github.io*, Jun 2023.

[18] Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.

[19] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.

[20] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2022. *URL https://arxiv. org/abs/2205.11916*, 2022.

[21] 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 *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2023.

[22] 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 *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530, 2023.

[23] Siyuan Huang, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *arXiv preprint arXiv:2305.11176*, 2023.

[24] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

[25] 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.

[26] 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.

[27] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of the International Conference on Machine Learning*, pages 9118–9147, 2022.

[28] 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.

[29] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. Tidybot: Personalized robot assistance with large language models. *arXiv preprint arXiv:2305.05658*, 2023.

[30] Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023.

[31] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[32] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the Conference on Robot Learning*, pages 2165–2183, 2023.

[33] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. PaLM-E: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[34] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[35] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[36] Eric Rohmer, Surya PN Singh, and Marc Freese. V-REP: A versatile and scalable robot simulation framework. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326, 2013.

[37] Stephen James, Marc Freese, and Andrew J Davison. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

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