Grounding Classical Task Planners via Vision-Language Models

Xiaohan Zhang[1], Yan Ding[1], Saeid Amiri[1], Hao Yang[2], Andy Kaminski[2], Chad Esselink[2], Shiqi Zhang[1]

***Abstract*— Classical planning systems have shown great advances in utilizing rule-based human knowledge to compute accurate plans for service robots, but they face challenges due to the strong assumptions of perfect perception and action executions. To tackle these challenges, one solution is to connect the symbolic states and actions generated by classical planners to the robot's sensory observations, thus closing the perception-action loop. This research proposes a visually-grounded planning framework, named TPVQA, which leverages Vision-Language Models (VLMs) to detect action failures and verify action affordances towards enabling successful plan execution. Results from quantitative experiments show that TPVQA surpasses competitive baselines from previous studies in task completion rate.**

I. Introduction

Classical planning frameworks such as those defined by Planning Domain Definition Language (PDDL) and Answer Set Programming (ASP) have been extensively utilized in planning and reasoning robot actions for long-horizon tasks [1]. Those classical planning systems are good at leveraging rule-based human knowledge to compute correct plans but suffer from the strong assumptions of perfect perception and action executions. For example, if the world model includes an apple on a table, classical planners assume that the robot will always locate the apple after reaching the table's location, and picking up the apple will deterministically result in it being in the robot's hand. These assumptions fail to consider dynamically changing environments and uncertain action outcomes, rendering it impractical for the robot to complete tasks by simply following computed plans in the real world.

To enable successful plan executions, classical planning systems are frequently accompanied by a plan monitoring system for linking the symbolic states and actions to robot sensory observations, where significant engineering efforts are needed. Fig. 1 illustrates the role of a plan monitoring system. Given the natural connection between planning symbols and human language, this paper investigates how pre-trained Vision-Language Models (VLMs) can assist the robot in realizing symbolic plans generated by classical planners, while avoiding the engineering efforts of checking the outcomes of each action. Specifically, we propose a vision-based symbolic planning framework, called TPVQA, that leverages VLMs to detect action failures and verify action affordances towards successful plan execution (Fig. 2). We take the advantage of the domain knowledge encoded in classical planners, including the actions defined by their effects and preconditions. By simply querying current observations

[1] Department of Computer Science, SUNY Binghamton
[2] Ford Motor Company

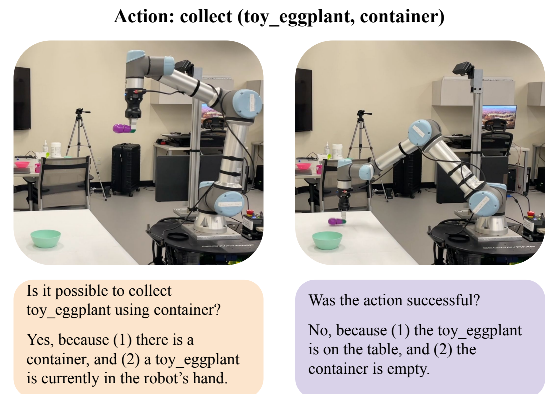


Fig. 1: To monitor plan executions on robots, it is required to answer the following questions: 1) *"Is it feasible to perform a particular action in the current state of the world?"* and 2) *"Was the action successfully executed, resulting in world transitions to the desired state?"*

against the action knowledge, similar to applying VLMs to Visual Question Answering (VQA) tasks, TPVQA can trigger the robot to repeat an unsuccessful action or call the symbolic planner to generate a new valid plan.

We conducted quantitative evaluations of TPVQA on an image dataset that consists of 20% realistic photos taken from home environments. The remaining images are augmented using diffusion models [2]. Experimental results demonstrate that TPVQA outperforms competitive baselines from the literature, achieving the highest task completion rate. Furthermore, we present an illustrative trial of deploying TPVQA on real robot hardware to perform object rearrangement tasks.

II. Related Work

A. Robot Planning with Classical Planners

Classical planning algorithms have found widespread application in robot systems. Recent classical planning systems designed for robotics commonly employ Planning Domain Description Language (PDDL) or Answer Set Programming (ASP) as the underlying action language for planners [3], [4], [5], [6]. Researchers have utilized classical planning algorithms for various robotic applications, including sequencing actions for a mobile robot on delivery tasks [7], reasoning about safe and efficient urban driving behaviors for autonomous vehicles [8], planning actions for a team of mobile robots [9], and completing service tasks in open-world scenarios [10]. Task and Motion Planning (TAMP), a hierarchical planning framework that combines classical planning in discrete spaces and robot motion planning in

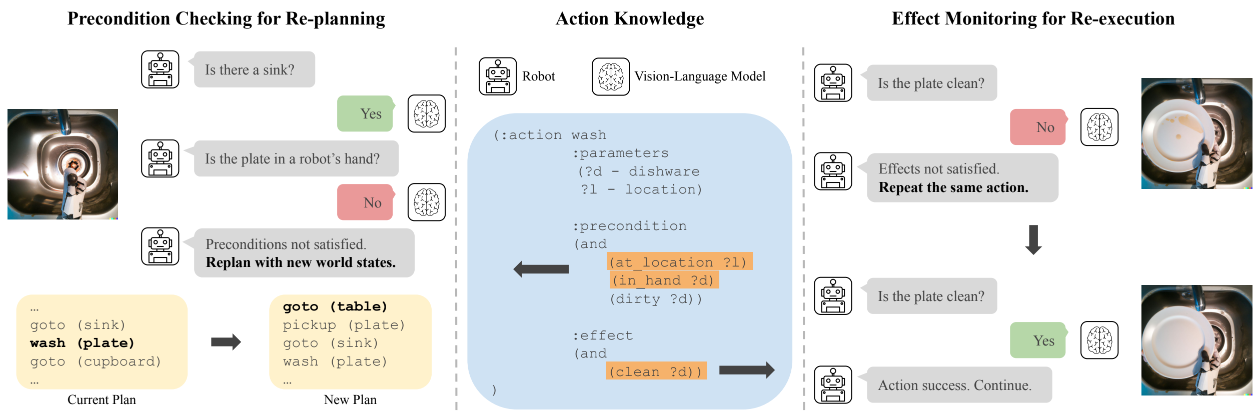


Fig. 2: Overview of TPVQA. By simply querying the robot's current observation against the action knowledge (i.e., effects and preconditions) as Visual Question Answering (VQA) tasks, TPVQA can trigger the robot to repeat an unsuccessful action, or call the classical planner to generate a new valid plan using an updated world state.

continuous space, has also shown great advances in robot long-horizon planning [11], [12]. Most classical planning algorithms that are designed for robot planning do not consider perception. Though some recent works have already shown that training vision-based models from robot sensory data can be effective in plan feasibility evaluation [13], [14], [15], [16], [17], [18], their methods did not tightly bond with language symbols which are the state representations for classical planning systems. We, on the other hand, propose TPVQA that uses VLMs to connect language (in classical planner) to robot perception.

B. Pre-trained Vision-Language Models in Robotics

Existing research has shown that large Vision-Language Models like CLIP [19] can be used in the robotics domain such as semantic scene understanding [20], effective open-ended agent learning [21], guiding robot navigation [22] and manipulation behaviors [23], [24]. Two recent works that are the most related to us are SuccessVQA [25] and PaLM-E [26]. SuccessVQA has investigated how VLMs enable robots to detect action outcomes and model action rewards. They also treat success/failure detections as VQA tasks, but did not consider affordances before action execution. PaLM-E is a large embodied VLM that is trained to predict robot action sequences as well as solve other downstream vision-language tasks. PaLM-E has demonstrated its effectiveness in both failure detection and affordance prediction. Different from their work, TPVQA uses classical planners for generating symbolic plans instead of solely relying on pre-trained models. That is because classical planning techniques are designed to ensure the generated plans are sound and complete. In addition, all the works that are listed in this section require additional training or fine-tuning VLMs in specific domains, but we study if and to what extent existing VLMs can help robot planning.

III. Background

In this section, we briefly summarize basic concepts in classical planning and Vision-Language Models, which serve as the two main building blocks of this research.

A. The Classical Planning Problem

Formally, the input of a planning problem P is defined by a tuple $\langle S, s^{init}, S^G, A, f \rangle$. S is a finite and discrete set of states used to describe the world's state (i.e., state space). We assume a factored state space such that each state $s \in S$ is defined by the values of a fixed set of variables. $s^{init} \in S$ is an initial world state. $S^G \subset S$ is a set of goal states. S^G are usually specified as a list of *goal conditions*, all of which must hold in a goal state. A is a set of symbolic actions. Actions are defined by their preconditions and effects. f is the underlying state transition function. State transitions are usually deterministic in classical planning problems but are not in real-world scenarios. A solution to a classical planning problem P is a symbolic plan π in the form of $\langle a_1, a_2, \dots, a_N \rangle$, such that the preconditions of a_1 hold in s^{init}, the preconditions of a_2 hold in the state that results from applying a_1, and so on, with the goal conditions all holding in the state that results after applying a_N.

B. Vision-Language Models for VQA Tasks

A general definition for VLMs is that they are models that combine both vision and language modalities. Most VLMs require encoders for both vision and language so as to train joint feature embeddings. One typical training strategy is by using Contrastive Learning [27]. Pre-trained VLMs have shown impressive capabilities in downstream tasks such as image captioning [28], open-vocabulary object detection [29], and visual question answering [30]. In this work, we relate robot actions with VQA queries for grounding long-horizon planning. We use the ViLBERT model [31] pre-trained on the VQA v2.0 dataset [32] which is publicly available in the AllenNLP platform [33].

IV. Method

This section presents our main contribution, TPVQA, that leverages VLMs to detect action failures and verify action affordances for enabling successful plan executions.

A. Precondition Checking for Re-planning

Before every action execution, TPVQA extracts knowledge about action preconditions from the planner's domain description. For instance, action `place_on(A, B)` has preconditions of `in_hand(A)` and `near(B)`, meaning that to place an object *A* on top of object *B*, the robot should first grasp *A* in hand and be located near object *B*. Then, we simply convert each action precondition into a natural language query by using some manually defined templates, such as *"Is object A in a robot's hand?"* and *"Is there an object B in the image?"* Paring each natural language query with the current observation from the robot's first-person view, we call the VLM to get answers indicating if the precondition is satisfied.

According to the results from the VLM, TPVQA will update the current state information in the classical planning system. Fig. 2 shows an example where the robot wants to `wash(plate)` but fails to detect "plate in a robot's hand" given the current image. Because the classical planner always assumes perfect action executions, it will incorrectly believe all previous actions are successful and the current world state includes `in_hand(plate)`. As a result, TPVQA will update the current state by removing `in_hand(plate)`. We provide the updated world state to the planner as the "new" initial state to re-generate a plan. In the above example, instead of `wash(plate)`, the robot will now take the action of `goto(table)` as it believes there is a plate on the table (according to the domain knowledge provided in the planning problem description).

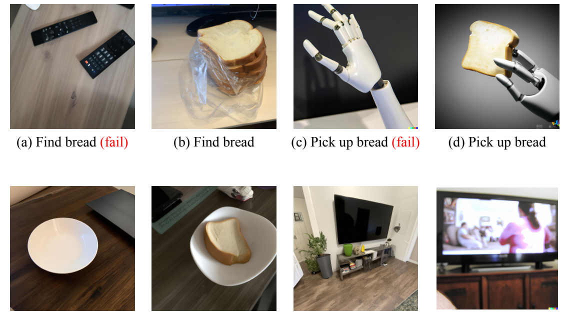
B. Effect Monitoring for Re-execution

After every action execution, TPVQA extracts knowledge about action effects from the planner's domain description. Similar to how TPVQA asks about preconditions, it queries action effects by using the VLM. If the effects are not satisfied, the robot will repeat the same action until it gets positive feedback from the VLM so as to continue the next action. Note that before re-trying each action, the robot will also need to check preconditions, because action failures frequently cause some preconditions to break in the real world. For example, failing to place an apple on the table might result in the apple falling to the ground, instead of still being in the robot's hand.

Remarks: A single action is usually defined by multiple preconditions and effects. VLMs, especially for those that are not trained using domain-specific data, frequently produce inaccurate answers that cause disagreements among the given preconditions (or effects). For instance, the VLM might answer "Yes" to both `on(apple, table)` and `in_hand(apple)` after the robot picks up an apple from the table. In this paper, we query the VLM about all the listed effects (preconditions), and determine to re-execute (re-plan) if the majority of them are not satisfied.

V. Experiments

We conduct extensive experiments to evaluate the performance of TPVQA comparing with baselines from the literature. Our hypothesis is that TPVQA produces the highest task completion rate because of its effectiveness in plan monitoring and online re-planning using perception. In the experiment, we consider three everyday tasks that are "clean_dishes", "serve_breakfast", and "eat_apple". Task descriptions are constructed using PDDL and symbolic plans are generated using the FAST-DOWNWARD planner[1], as shown in TABLE I.



(a) Find bread (fail) (b) Find bread (c) Pick up bread (fail) (d) Pick up bread

(e) Find plate (f) Place bread on plate (g) Find TV (h) Turn on TV

Fig. 3: An example trial for "serve_breakfast" task sampled from the simulator.

clean_dishes	serve_breakfast	eat_apple
Step 1: Find plate Step 2: Pick up plate Step 3: Find sink Step 4: Wash plate	Step 1: Find bread Step 2: Pick up bread Step 3: Find plate Step 4: Place bread on plate Step 5: Find TV Step 6: Turn on TV	Step 1: Find fridge Step 2: Open fridge Step 3: Find apple Step 4: Pick up apple Step 5: Find knife Step 6: Pick up knife Step 6: Cut into half apple

TABLE I: Symbolic plans computed for the tasks.

A. Simulator with Diffusion Models

To quantitatively evaluate the performance of TPVQA in dealing with imperfect perception and uncertain action outcomes, we build a simulator using web-scale diffusion models. We first took images from real environments and use the image variation API provided by DALL-E [2] to augment the original dataset. For a small portion of the actions for which real photos are difficult to get, such as a robot washing a plate, we manually design prompts as inputs to DALL-E. Each action is paired with 10 successful observations and 10 failed ones. Overall, our image dataset consists of 20% real photos, 60% images from real photo variations, and 20% images directly generated from text prompts. Fig. 3 shows example images from our dataset.

At each time step, an observation for the current action is sampled from the dataset. We assume that there is a probability of 25% that an action may fail which will result in a failed action observation. We also assume there is another 25% chance that a failed action may cause changes to previous states. For instance, when the robot fails on the action `cutintohalf(apple)`, there is a chance that the apple (or the knife) is not in the robot's hand anymore. To model this uncertainty, we re-sample one of the previous

[1]See `https://www.fast-downward.org/` for the details on the FAST-DOWNWARD software. We use the implementation from `https://github.com/aibasel/downward`.

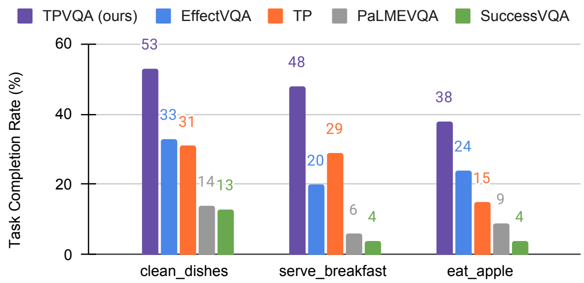


Fig. 4: Task completion rates of TPVQA and four baselines evaluated over three tasks.

observations to let the robot estimate the current world state. The robot needs to successfully execute all the actions so as to complete the task, leading the system to the desired goal state.

B. Baselines

TPVQA is compared with the following four baselines:

- EffectVQA: An ablative version of ours where VLMs are only used for action effect monitoring.
- TP: A task planning baseline without perception.
- PaLMEVQA: PaLM-E [26] is robust to most of the vision-language downstream tasks. Among those, we are more interested in affordance prediction and failure detection. To this end, PaLMEVQA is a baseline that is designed with prompts provided in the original PaLM-E paper, which are *"Is it possible to <action> here?"* and *"Was <action> successful?"*
- SuccessVQA [25]: We use the same query provided in their paper, which is *"Did the robot successfully <action>?"* SuccessVQA does not consider affordance.

Note that neither PaLM-E nor Flamingo [34] (as used in the original SuccessVQA paper) is open-sourced, so we use the same VLM as ours [31] for implementing their corresponding baselines. As discussed in Section II-B, both PaLM-E and SuccessVQA are trained on robotics data, but all evaluations in this paper do not involve any altering for the VLM itself.

C. Results

Fig. 4 presents the main experimental results. We observe that TPVQA consistently outperforms baselines in task completion rate, which supports our hypothesis. As the number of required action steps of tasks increases, the success rates of all the methods decrease as expected. By considering action knowledge (i.e., preconditions and effects), TPVQA and EffectVQA are significantly better than others, especially the ones (PaLMEVQA and SuccessVQA) that only query about actions by their names. We can also tell that methods additionally considering action affordances (TPVQA and PaLMEVQA) perform better than the methods that only detect action failures (EffectVQA and SuccessVQA).

Another interesting finding is that TP, a baseline that does not include any perception, produces a higher success rate than PaLMEVQA and SuccessVQA, which are two other baselines that are capable of interacting with VLMs. That is because false positives and false negatives from VLMs will greatly impact the plan execution, easily leading the robot to failure cases. TABLE II shows that when querying about action knowledge, the VLM is more accurate in failure detection and affordance prediction. If directly querying about action names, prediction accuracies for most of the actions are lower than a random guess, which is why SuccessVQA and PaLMEVQA perform poorly. A straightforward example is when detecting if the robot has successfully washed the plate, instead of asking *"Was wash plate successful?"*, TPVQA will query *"Is the plate clean?"*. We observe that the latter type of queries (ours) is easier for VLMs to understand.

Query	Accuracy		
	clean_dishes	serve_breakfast	eat_apple
Action Pres.	**0.63**	**0.53**	**0.70**
Is <action> possible?	0.58	0.33	0.43
Action Effs.	**0.79**	**0.60**	**0.71**
Was <action> successful?	0.45	0.30	0.47

TABLE II: VQA accuracies on different querying strategies.

D. Real-Robot Deployment

We also deployed TPVQA on real robot hardware to perform object rearrangement tasks (Fig. 5), where the goal is to "collect" toys using a container and place them in the red area. Our real-robot setup includes a UR5e Arm with a Hand-E gripper mounted on a Segway base, and an overhead RGB-D camera (relatively fixed to the robot) for perception. Please refer to Appendix for more details.

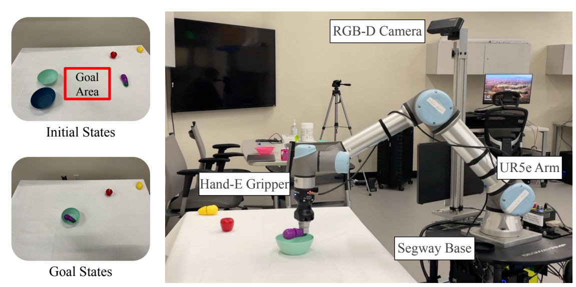


Fig. 5: Real robot demonstration for TPVQA.

VI. Conclusion and Future Work

In this paper, we investigate robot classical planning with pre-trained VLMs. We propose TPVQA that triggers re-planning using precondition checking and re-execution using effect monitoring. By doing a set of experiments on robots working on everyday tasks, we demonstrate that TPVQA is able to provide more successful plan executions than baselines. For future work, we would like to evaluate our method using more tasks, potentially those from existing benchmarks such as ActivityPrograms [35]. In addition to using images generated by diffusion models for evaluation, it might be possible to use datasets such as EGO4D [36]. To improve the overall task completion rate, one way is to collect task-specific data and finetune the pre-trained model. It will be also interesting to develop methods that can handle inconsistent inferences from VLMs.

References

[1] M. Ghallab, D. Nau, and P. Traverso, *Automated planning and acting*. Cambridge University Press, 2016.

[2] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8821–8831.

[3] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, and P. Stone, "Task planning in robotics: an empirical comparison of pddl-and asp-based systems," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, pp. 363–373, 2019.

[4] G. Brewka, T. Eiter, and M. Truszczyński, "Answer set programming at a glance," *Communications of the ACM*, vol. 54, no. 12, pp. 92–103, 2011.

[5] V. Lifschitz, "Answer set programming and plan generation," *Artificial Intelligence*, vol. 138, no. 1-2, pp. 39–54, 2002.

[6] M. Fox and D. Long, "Pddl2. 1: An extension to pddl for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.

[7] S. Zhang, F. Yang, P. Khandelwal, and P. Stone, "Mobile robot planning using action language bc with an abstraction hierarchy," in *International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer, 2015, pp. 502–516.

[8] Y. Ding, X. Zhang, X. Zhan, and S. Zhang, "Task-motion planning for safe and efficient urban driving," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[9] Y. Jiang, H. Yedidsion, S. Zhang, G. Sharon, and P. Stone, "Multi-robot planning with conflicts and synergies," *Autonomous Robots*, vol. 43, no. 8, pp. 2011–2032, 2019.

[10] Y. Ding, X. Zhang, S. Amiri, N. Cao, H. Yang, C. Esselink, and S. Zhang, "Robot task planning and situation handling in open worlds," *arXiv preprint arXiv:2210.01287*, 2022.

[11] F. Lagriffoul, N. T. Dantam, C. Garrett, A. Akbari, S. Srivastava, and L. E. Kavraki, "Platform-independent benchmarks for task and motion planning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3765–3772, 2018.

[12] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[13] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, "Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6541–6548.

[14] X. Zhang, Y. Zhu, Y. Ding, Y. Zhu, P. Stone, and S. Zhang, "Visually grounded task and motion planning for mobile manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1925–1931.

[15] D. Driess, J.-S. Ha, and M. Toussaint, "Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image," *arXiv preprint arXiv:2006.05398*, 2020.

[16] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint, "Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9563–9569.

[17] A. M. Wells, N. T. Dantam, A. Shrivastava, and L. E. Kavraki, "Learning feasibility for task and motion planning in tabletop environments," *IEEE robotics and automation letters*, vol. 4, no. 2, pp. 1255–1262, 2019.

[18] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," *arXiv preprint arXiv:2303.06247*, 2023.

[19] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[20] H. Ha and S. Song, "Semantic abstraction: Open-world 3d scene understanding from 2d vision-language models," in *Conference on Robot Learning*, 2022.

[21] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar, "Minedojo: Building open-ended embodied agents with internet-scale knowledge," *arXiv preprint arXiv:2206.08853*, 2022.

[22] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam, "Clip-fields: Weakly supervised semantic fields for robotic memory," *arXiv preprint arXiv: Arxiv-2210.05663*, 2022.

[23] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[24] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, B. Zitkovich, F. Xia, C. Finn, and K. Hausman, "Open-world object manipulation using pre-trained vision-language model," in *arXiv preprint*, 2023.

[25] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi, "Vision-language models as success detectors," *arXiv preprint arXiv:2303.07280*, 2023.

[26] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[27] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[28] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[29] 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 preprint arXiv:2205.06230*, 2022.

[30] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2425–2433.

[31] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," *Advances in neural information processing systems*, vol. 32, 2019.

[32] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the v in vqa matter: Elevating the role of image understanding in visual question answering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6904–6913.

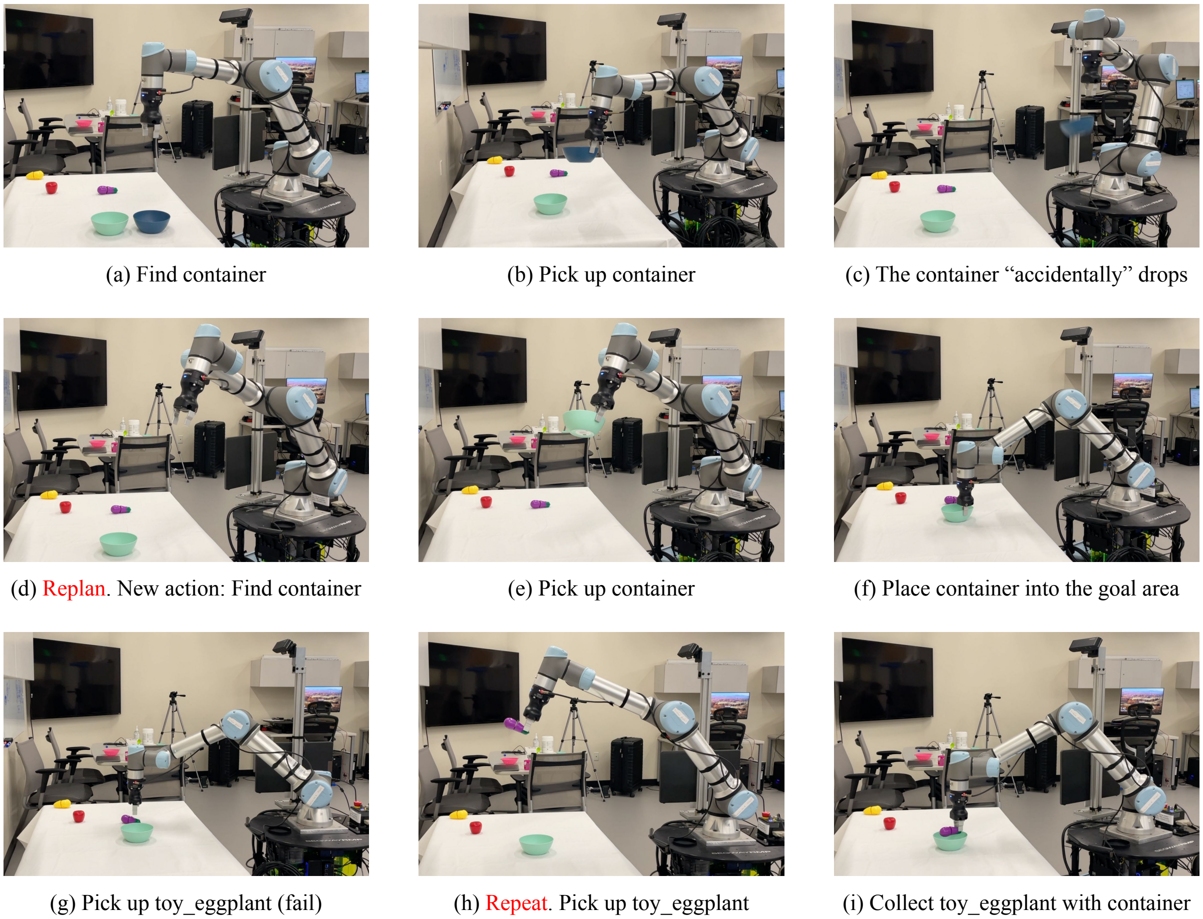
[33] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer, "Allennlp: A deep semantic natural language processing platform," *arXiv preprint arXiv:1803.07640*, 2018.

[34] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 23 716–23 736, 2022.

[35] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.

[36] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu *et al.*, "Ego4d: Around the world in 3,000 hours of egocentric video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 995–19 012.

[37] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *arXiv preprint arXiv:1804.05172*, 2018.



(a) Find container
(b) Pick up container
(c) The container "accidentally" drops
(d) Replan. New action: Find container
(e) Pick up container
(f) Place container into the goal area
(g) Pick up toy_eggplant (fail)
(h) Repeat. Pick up toy_eggplant
(i) Collect toy_eggplant with container

Fig. 6: Screenshots showing the full demonstration trial of TPVQA as applied to a real robot.

APPENDIX

Fig. 6 shows a sequence of screenshots of a real robot using TPVQA on object rearrangement tasks. The goal is to "collect" toys using a container and place them in the middle of the table (i.e., goal area). We assume that the robot has a predefined set of skills, including `pick`, `place`, and `find`. `Pick` and `place` actions are implemented using GG-CNN [37], and `find` action simply uses base rotation for capturing tabletop images from different angles.

Given the task description, the robot first decided to execute "find container" and "pick up container". These two actions were successfully executed as shown in Fig. 6(a), 6(b). When the robot was preparing the next action (i.e., "Place container into the goal area"), the blue container accidentally dropped from the robot's gripper to the ground (Fig. 6(c)). Instead of directly executing the next action, TPVQA enabled the robot to check preconditions by querying the VLM *"Is the container in a robot's hand?"* After receiving negative feedback from the VLM, TPVQA updated the world state by removing `in_hand(container)` and called the planner to generate a new plan that started the task again by finding another container (Fig. 6(d)). Then the robot picked up the cyan container and placed it in the middle of the table as shown in Fig. 6(e), 6(f). The subsequent actions in the plan were to find and pick up a toy, but the `pick` action failed (Fig. 6(g)). TPVQA managed to detect the failure by querying 1) *"Is there a toy_eggplant on the table?"*, and 2) *"Is the toy_eggplant in a robot's hand?"*, and receiving Yes and No answers respectively. As a result, our system suggested the robot repeat the `pick` action again (Fig. 6(h)). Finally, the robot successfully collected the toy by putting it into the cyan container that was previously placed in the goal area (Fig. 6(i)).