Plan Diffuser: Grounding LLM Planners with Diffusion Models for Robotic Manipulation

S P Sharan* spsharan@utexas.edu

Ruihan Zhao* ruihan.zhao@utexas.edu

Ufuk Topcu utopcu@utexas.edu

Zhangyang Wang atlaswang@utexas.edu

Sandeep Chinchali sandeepc@utexas.edu

The University of Texas at Austin

Abstract: Embodied AI is progressively exploring large language models (LLMs) for effective planning in robotics. Recent advancements in embodied AI have enabled LLMs to deconstruct a visual observation and a high-level goal prompt into executable sub-tasks. However, these existing methods often perform planning entirely based on the initial state of the environment, leading to a weakened grounding when generating longer plans. Some recent directions of research explore closing the loop through the incorporation of environmental feedback in the form of language. Unlike these methods, we introduce Plan Diffuser, a novel "closed-loop" approach for step-by-step planning with visual feedback accompanied at each step of the loop. Specifically, our method autoregressively employs an LLM to generate single-step text subgoals and a diffusion model to translate these into visual subgoals which are used for subsequent planning. Finally, a goal-conditioned policy capable of realizing these sub-goal images into robotic control actions executes them. Comprehensive evaluations on the Ravens benchmark suite reveal that Plan Diffuser surpasses state-of-the-art methods, particularly in long-horizon tasks. Furthermore, our approach demonstrates robust generalization capabilities in out-of-distribution scenarios – handling unseen colors, objects, and increased task complexity with ease.

Keywords: Closed Loop Planning, Large Language Models, Diffusion Models

1 Introduction

Embodied AI is emerging as a pivotal facet of recent robotics, centralizing the role of integrating perception, cognitive reasoning, and control [1, 2]. Robotic manipulation transcends mere action mapping from raw visuals and delves deep into understanding contextual cues from the environment. Consider the intricate task of building a structure based on high-level language instructions. Here, a robot is not just executing actions but engaging in a series of cognitive processes: perceiving and distinguishing different elements, understanding the objective/goal, and coherently weaving these insights to plan and manipulate objects into desired configurations. Such tasks necessitate a robust grounding for cognitive reasoning – encompassing scene interpretation, comprehensive goal understanding, and the synthesis of these elements for effective contextual planning.

The emergence of large vision and language models has sparked significant interest in the integration of these models within the field of Embodied AI. CLIPort [3] enables prompt-conditioned table-top robotic manipulation by using a joint vision-language model to map image observations and task prompts to affordances. More recently [4, 5, 6] harness the planning capability of LLMs by building

*These authors contributed equally to this work

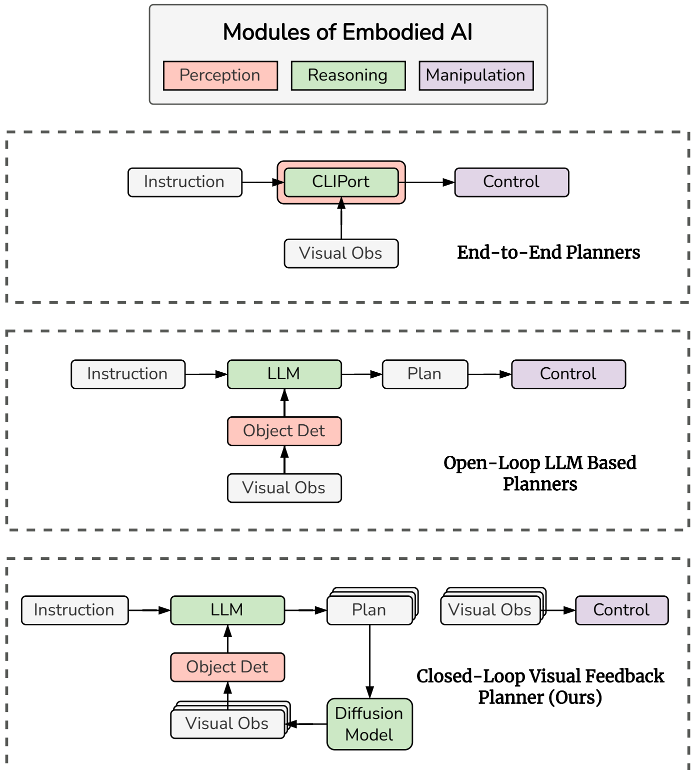


Figure 1: Plan Diffuser is a "closed-loop" planner that actively conditions upon the visual state of the environment throughout the planning process e unlike existing methods for robotic manipulation. This continuous interaction with intermediate states generated through a diffusion model enhances the system's contextual awareness and strengthens its grounding.

a text-based wrapper around the environment and deconstructing the high-level goal prompt into executable sub-tasks.

However, each approach grapples with distinct challenges. End-to-end approaches of mapping of observations to affordance maps as seen in CLIPort [3] unifies all three roles – perception, cognitive reasoning, and manipulation within a single model. This entangles the learning of domain-specific control policies and domain-agnostic vision-language grounding [7]. Thus, they are often prone to poor generalization, overfitting, and need for extensive task-specific training. On the other hand, LLM-based approaches successfully separate perception (e.g. object detectors), reasoning, and manipulation (e.g. pre-trained controllers), but they usually plan entirely from the environment's initial state. We label this the "open-loop" approach because planning occurs in a *single-shot* (prompt $\rightarrow$ plan), relying solely on the initial observation and blind to intermediate future states.

To this end, we propose **Plan Diffuser**, a novel "closed-loop" planning technique that actively considers the visual state of the environment throughout the planning process. By capturing the intricate spatial layout of objects through visual feedback, our algorithm is better grounded as it (1) has the ability for contextually relevant cognition, and (2) relays intermediate observations back to the planner. At the heart of our approach are two core components: an LLM, which serves as the primary planner, and a diffusion model, responsible for generating image sub-goals and feeding back into the planner for the next step. As illustrated in Figure 1, these two elements operate in an auto-regressive manner, ensuring that our method remains solidly grounded to the evolving environment over long horizon tasks. Additionally, Plan Diffuser exhibits enhanced generalization in more challenging tasks that span longer durations, involve unseen colors, or a larger number of objects. After the generation of sub-goal images, a TransporterNet [8] style goal-conditioned policy executes the plan.

Furthermore, our approach boosts the safety of embodied agents as we generate sub-goal images prior to their execution. By involving human-in-the-loop visualization and verification, it allows plans to be validated before deployment on real robots. Because our methodology leverages a prompt-based image-to-image diffusion model to synthesize these images, it also facilitates the manual editing of any erroneous sub-goals through further prompting prior to deployment. This sharply contrasts with methods like Inner Monologue [9] and LILAC [10] which make real-time corrections during execution. Although more agile, these methods might produce risky behaviors in safety-critical scenarios.

In summary, our contributions are as follows:

1. We propose **Plan Diffuser**, a novel method to tackle long-horizon robotic manipulation tasks using a "closed-loop" planner that actively conditions upon the visual state of the environment throughout the planning process. Plan Diffuser consists of an LLM as the planner and a diffusion model responsible for generating the intermediate states. Their autoregressive collaboration, as shown in Figure 1, enhances the system's contextual awareness and solidifies its grounding.
2. Plan Diffuser decomposes high-level textual goals into actionable atomic image sub-goals, significantly shortening the horizon of the image-based robot controller, and sidestepping the pitfalls commonly associated with end-to-end and text-only strategies.
3. Plan Diffuser outperforms state-of-the-art methods in long-horizon tasks in the Ravens environment, without the need for task-specific models, as is often mandated in current methods. We further demonstrate the superior generalization capability of Plan Diffuser in terms of unseen colors, objects, and increased task complexities.

2 Related Works

In this section, we discuss recent progress in robot learning leveraging the power of large language models and computer vision techniques.

Language-based reasoning for robotics. Aside from natural language processing, large language models demonstrate strong capabilities in reasoning about and planning in the physical world. When code-writing LLMs are provided API access to robot perception and action subroutines, they can program robots to accomplish tasks such as tabletop manipulation and mobile manipulation [4, 6]. Furthermore, LLM's summarization capability enables the learning of user preferences from a few prior examples, thereby disambiguating among multiple potential object arrangements [11]. However, additional grounding is usually needed to generate feasible plans considering the current configuration of a non-stationary environment. To provide environment grounding directly, prompts to the LLM can be structured with program-like specifications of environment configurations and feasible actions [12]. Alternatively, feedback from the environment can also be encoded in language and used for future plans in a closed-loop fashion [9]. Finally, given a collection of robot skills, their utility and feasibility can be evaluated in the current environment to select the best plan [5].

Vision-based robotic manipulation. LLM-based planners usually rely on perception subroutines such as object detection and segmentation, potentially limiting their application in unstructured or complex environments. On the other hand, prior works have explored various ways to learn manipulation skills directly from pixels. For example, TransporterNet [8] models table-top manipulations as a problem to rearrange deep visual features, leveraging spacial symmetries. As an imitation learning method, it is shown to learn faster and generalize better in a wide range of tasks. In the domain of reinforcement learning, techniques including data augmentation and contrastive learning help train good image feature extractors to enable efficient model-free RL [13]. On the other hand, MoDem [14] leverages demonstrations in model-based RL to achieve superior performance in sparse-reward data-limited table-top and dexterous manipulation tasks.

Multi-modal robot learning. To combine the reasoning power of large language models and reach sensor information, recent works on multi-modal robot learning explore ways to bridge language and various perception models. CLIPort [3] improves upon TransporterNet and trains language-conditioned visual imitation learning agents to harvest the semantic understanding from a pre-trained CLIP vision-language model [15]. Instruct2Act [16] provides LLMs access to foundation models including Segment Anything Model (SAM) and CLIP to accurately locate and classify manipulation objects [17]. PaLM-E [18] employs multi-modal architectures to learn robot actions directly from a dataset including images, robot state estimations, and text inputs. RT-2 [19] shows that by also training on internet-scale data, vision-language-action models benefit from better generalization to novel objects and commands.

In conclusion, there remain open challenges with existing approaches. Language-based planners usually assume access to ground truth object positions and masks, as well as high-level pick-place primitives [4, 6, 20], hindering their practicality in real-world deployment. Alternatively, learning imaged-based control from scratch can require a lot of data and lack generalization [8, 3, 13, 14]. Finally, training generalizable multi-modal embodied agents might demand tremendous data, annotations, and compute resources [18, 19].

3 Methodology

As illustrated in our teaser (Figure 1), Plan Diffuser begins by accepting an initial visual observation of the environment along with a high-level goal described in natural language as inputs. It then interprets these inputs and generates sub-goal images depicting the necessary steps for task completion. Finally, a goal-conditioned policy translates these sub-goal images into robotic commands. In this section, we formally describe the task at hand and provide an explanation of the pipeline in Section 3.1. We include the implementation details in Section 4.4.

3.1 Functional Modules

We formulate the robotic manipulation problem as an atomic plan decomposition task. These atomic plans are sub-goal images executable by a goal-conditioned policy. In particular, given the high-level task prompt $\mathcal{P}$ in natural language, and the initial RGBD observations of the environment $\mathcal{I}$, our method aims to generate sub-goal images, $\mathcal{P} \times \mathcal{I} \rightarrow \mathcal{G}_I = \{g_1, g_2, ..., g_n\}$. A goal-conditioned policy translates these into robotic control $\texttt{GCP} : \mathcal{G}_I \rightarrow \mathcal{C}$. As such, Plan Diffuser's pipeline contains the following three main models:

1. **The Object Detector** (`OD`) is an open-vocabulary object detection model. It is responsible for generating the bounding-box coordinates of relevant objects in the form of natural language text snippet s, given an image observation g.

$$s_i = \texttt{OD}\,(g_i) \tag{1}$$

2. **The LLM Planner** (`LLM`) is tasked to generate a text-based instruction t as the next atomic action, given the high-level goal prompt $\mathcal{P}$ as well as the `OD`-parsed state s of the current environment configuration.

$$t_{i+1},\,\text{done} = \texttt{LLM}\,(\mathcal{P},\, s_i) \tag{2}$$

3. **The Sub-Goal Generator** (`SG`) is a prompt-based image editing diffusion model. The job of `SG` is to take the current observation g_i and synthesize the next-step image observation g_{i+1} assuming the text instruction t_{i+1} is successfully executed.

$$g_{i+1} = \texttt{SG}\,(t_{i+1},\, g_i) \tag{3}$$

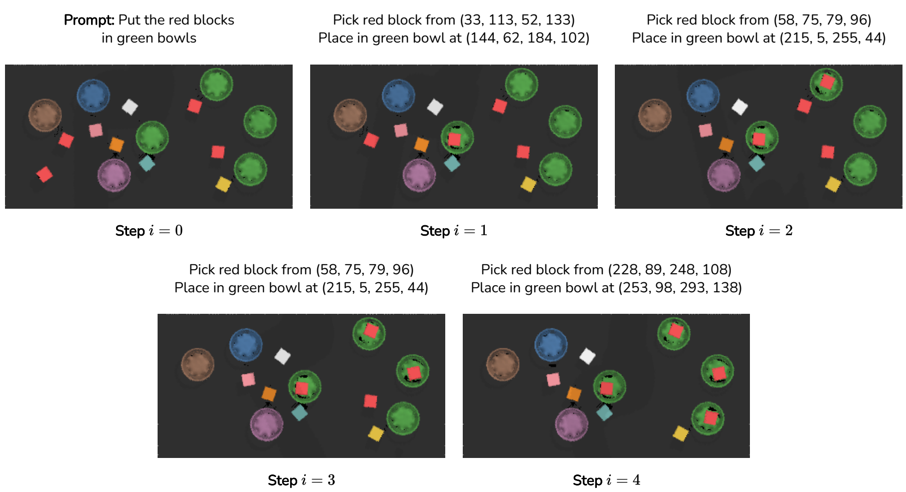
The pipeline starts with $g_0 = \mathcal{I}$, and repeats $\forall\, i \in \{0, 1, 2, ..., n\}$ until the LLM planner signals the conclusion of execution. The pseudo-code of our method is shown in Algorithm 1.

Algorithm 1: Plan Diffuser

Input: High level task prompt $\mathcal{P}$, initial RGBD configuration $\mathcal{I}$
Prompts: System prompt $\mathcal{P}_{sys}$, Few-shot example prompts $\mathcal{P}_{eg}$
Output: Robotic control $\mathcal{C}$

begin
  $g_0 \leftarrow \mathcal{I}$ `// The first sub-goal image is the initial observation`
  $i \leftarrow 0$
  done $\leftarrow$ False
  while done $\neq$ True **do**
    $s_i = \texttt{OD}\,(g_i)$ `// Object Detector`
    t_{i+1}, done $= \texttt{LLM}\,(\mathcal{P}_{sys},\, \mathcal{P}_{eg},\, s_i)$ `// LLM Planner`
    $g_{i+1} = \texttt{SG}\,(t_{i+1},\, g_i)$ `// Sub-Goal Generator`
    $i \leftarrow i + 1$
  $\mathcal{G}_I \leftarrow \{g_1, g_2, ..., g_n\}$ `// Sub-goal images leading to goal completion`
  $\mathcal{C} = \texttt{GCP}\,(\mathcal{G}_I)$ `// Execute with a goal-conditioned policy`

(a) put-blocks-in-bowls-unseen-colors with horizon $= 3$



(b) put-blocks-in-bowls-unseen-colors with horizon $= 5$

Figure 2: Examples of text-based instructions and sub-goals generated by Plan Diffuser

3.2 Plan Diffuser

In this section, we dive into the details of our methodology. We go through Algorithm 1 and refer to the example in Figure 2a to aid our explanation. Here, Plan Diffuser takes the high-level goal prompt "Put the white blocks in brown bowls" ($\mathcal{P}$) and the top view image depicting the arrangement of blocks and bowls in the environment ($\mathcal{I}$) as inputs. As the first step, an open-vocabulary object detector performs inference on $\mathcal{I}$ and generates a list of detected objects, their labels, and corresponding bounding boxes. The following is the output generated for this scene:

The textual representation of a scene s_i obtained from the Object Detector

```
1.  brown bowl at (118, 95, 157, 135)
2.  pink block at (172, 137, 192, 158)
3.  white block at (229, 110, 249, 131)
...
9.  red block at (42, 80, 23, 100)
10.  green block at (176, 40, 196, 61)
```

The textual conversion of the scene is performed to produce an input to the LLM. As depicted in L7 of Algorithm 1, the LLM takes in 3 prompts concatenated together. Firstly the system prompt $\mathcal{P}_{sys}$ provides the LLM an understanding of the task, the environment, its rules, and available actions. Secondly, for few-shot reasoning [21], a few example conversations of how the LLM is expected to reason and respond is provided in $\mathcal{P}_{eg}$. Lastly, the textual representation of the scene is appended. We provide these complete prompts in Appendix A. In the case of Figure 2a, the LLM identifies that there are two white blocks and two brown bowls. Thus, the first step would be to move one of the white blocks to one of the brown bowls (as seen in the center caption). The complete output given by the LLM is as follows:

The reasoning and the first-step text-based instruction generated by the LLM

```
reasoning:  there are two white blocks and two brown bowls.  both the white
blocks are outside the brown bowls.  therefore they have to be moved into the
brown bowls as per the goal.  we first move one of the white blocks to the
brown bowl.
pick:  3.  white block:  (229, 110, 249, 131)
place:  1.  brown bowl:  (118, 95, 157, 135)
```

An output parser decomposes the text into its constituent pick/place objects and their coordinates. The LLM also outputs a "`done`" flag which denotes when the plan is complete. The subgoal generator `SG` takes in the mask of the pick object (which was generated by the object detector) and through inpainting using a diffusion model matches the background texture. The object is then ”placed” at the target coordinates using the same mask and inpainting technique this time to infill it with a block of the same color. This imagined sub-goal is shown in the center image in Figure 2a.

This process is repeated till the "`done`" flag is reported as True.

4 Experimental Setup

4.1 Benchmark Environment

To assess Plan Diffuser's performance, we utilize tasks from the CLIPort [3] benchmark, which is rooted in the benchmark suite known as Ravens [8] and built upon the Pybullet framework [22]. Central to this benchmark are language-conditioned tabletop manipulation tasks. These tasks present an initial state observation alongside a high-level goal prompt, which guides the subsequent robotic manipulations (refer Figure 2).

The robotic arm employed in these tasks is the Universal Robot UR5e, equipped with a suction gripper end-effector. Task instances are defined by sampling a varied set of objects and their associated attributes, including poses, colors, sizes, and object categories. For the scope of this study, we specifically focus on two long-horizon, language-conditioned tasks with high-level goals: "put-blocks-in-bowls" and "packing-google-objects-group". Both tasks come in two variants – seen (or) unseen. The distinction arises from differences in object colors and types between the training vs. testing phases.

4.2 Metrics

Aligned with the Ravens benchmark [8], our evaluation employs a scoring scale ranging from 0 (complete failure) to 100 (complete success). This scoring system allows for partial credit based on the extent of task completion. For instance, if a task necessitates the movement of 4 blocks and the agent successfully relocates only 2, a score of 50.0 (or $2/4$) would be assigned. The agent continues its interaction with the environment until either an oracle indicates the task's completion or a preset timeout is reached.

4.3 Baseline Methods

We evaluate our approach against two prominent baseline methods: TransporterNet [8] and CLIPort [3]. TransporterNet primarily operates by rearranging deep features, allowing it to infer spatial displacements from visual input. This mechanism effectively parameterizes robot actions. Building upon the foundation laid by TransporterNet, CLIPort is a language-conditioned imitation learning agent that integrates the broad semantic understanding (what) provided by CLIP [15] with the spatial precision (where) offered by Transporter [8].

For a comprehensive comparison, we assess against CLIPort in two distinct settings: the "single" setting, where the model is trained and evaluated on a single task individually; and the "multi" setting, where the model undergoes training on all tasks concurrently, but each task's evaluation is conducted independently. Our experiments involve 1,000 training demonstrations for each task and are evaluated using 100 test demonstrations.

4.4 Implementation Details

Building on our earlier discussion in Section 3.1, Plan Diffuser incorporates several modules, each tailored for specific roles in robotic manipulation tasks. We employ `gpt-3.5-turbo` for our LLM planning, accessed via LangChain [23], for prompt engineering. The LLM instructions contain the setup prompt about the manipulation task, available actions, and the high-level goal. By including a few planning examples, we leverage few-shot prompting [21], aiming to heighten planning accuracy. For sub-goal image synthesis we adopt ControlNet backed by StableDiffusion [24, 25]. Using masks from the object detector, we perform in-painting "pick" locations with their background, and regenerate them at their "place" locations, effectively simulating movement. As these steps permit visualization of intermediate outputs, and manual adjustment of masks and prompts to edit behavior, preemptive replanning can be performed to correct potential errors. ViLD [26] is used as the open vocabulary object detector.

5 Results

Building upon the experimental setup, we turn our attention to the empirical evaluation of our proposed approach on the Ravens benchmark suite [8]. Through this section, we gauge the efficacy of our method and validate the significance of our proposed contributions via a series of ablations. Our experiments are conducted to answer the following pivotal questions:

1. Does closed-loop planning with visual feedback actually enhance long-horizon performance?
2. In scenarios demanding out-of-distribution generalization and increased task complexities, does stronger grounding play a beneficial role?
3. To what extent can we attribute performance gains to step-by-step planning and the inclusion of vision-in-the-loop feedback?

Table 1: Performance comparison of Plan Diffuser and existing methods on the Ravens benchmark. Plan Diffuser shows consistent performance across both "seen" and "unseen" scenarios. Numbers represent task completion accuracy (the higher the better).

Task	put-blocks-in-bowls seen-colors		put-blocks-in-bowls unseen-colors		packing-seen-google objects-group		packing-unseen-google objects-group	
#demonstrations	100	1000	100	1000	100	1000	100	1000
TransporterNet	62.7	64.7	14.8	18.7	61.5	59.9	49.8	52.0
CLIPort (single)	92.5	100	37.3	25.0	84.1	94.0	78.4	81.5
CLIPort (multi)	**99.5**	**100**	55.3	45.8	89.6	88.3	78.4	80.3
Plan Diffuser (ours)	98.5		**83.4**		**94.8**		**85.5**	

Table 2: Generalization across longer horizons, unseen colors, added distractors on "put-blocks-in-bowls-seen-colors"

Modification	Longer Horizon			Unseen† Colors	#Blocks		
Value	3†	5	10		5†	7	9
TransporterNet	64.7	58.81	40.58	18.7	64.7	59.55	43.77
CLIPort (single)	100	71.41	55.82	25.0	100	73.73	59.33
CLIPort (multi)	**100**	74.29	62.44	45.8	**100**	78.57	63.98
Plan Diffuser (ours)	98.5	**91.31**	**83.70**	**83.4**	98.5	**91.95**	**84.58**

† This corresponds to the standard setting as measured in Table 1.

5.1 Comparison with Existing Methods

As outlined in Section 4.1, we assess our method on the Ravens benchmark [8], mirroring the setup used by CLIPort [3]. Our results in Table 1 show that Plan Diffuser consistently outperforms existing methods in these long-horizon tasks. CLIPort blends the task of perception, reasoning, and manipulation into one single model and we believe this might hinder learning. Specifically, we believe performance is inhibited due to entanglement between learning a domain/task-specific control policy and a domain/task-agnostic vision-language grounding. Our approach avoids this by using separate role-specific modules (refer Figure 1). Notably, Plan Diffuser maintains strong performance in both familiar ("seen") and new ("unseen") tasks. This negates the need for task-specific models or extensive expert demonstration collection for each new task. For a detailed breakdown of Plan Diffuser's components and their impact, please refer to our ablation in Section 5.3.

5.2 Grounding in OOD Generalization and Task Complexity

While CLIPort offers a valuable benchmark, it may not capture the full challenges of real-world tabletop tasks. We've intensified the "put-blocks-in-bowls-seen-colors" task by increasing its complexity in three ways: longer horizon (more pick-place operations for completion), adding blocks of unseen colors, and introducing more distractor blocks (task-irrelevant objects). This both tests the limits of existing planners and also deepens our understanding with their shortcomings in various OOD settings, thus paving the way for further research. Our results in Section 5.2 show that while traditional methods falter with these added challenges, Plan Diffuser remains resilient. Its closed-loop planning is especially adept at handling longer tasks, and the decoupled perception-reasoning modules effectively adapt to new colors and distractions.

5.3 Ablation: Step-by-Step Planning vs. Vision-in-the-Loop Feedback

LLMs have been recognized as an effective planning tool for robotic tasks [4, 6, 9, 5]. In this ablation, we examine whether enhancing LLM-based planners with closed-loop visual feedback leads to improved outcomes. We begin with access to an object detector (OD) and high-level API environment control, akin to [4]. We pick the "put-blocks-in-bowls-unseen-colors" task and perform

Table 3: Task accuracy progression from single-shot LLM planning (as [4]), to step-by-step prompting (as [27]), and finally to our proposed method with grounded visual feedback

Step-by-Step Planning	Vision Feedback	Task Acc. (↑)
✗	-	72.29
✓	✗	75.63
✓	✓	**83.4**

ablations to verify the benefits of our method. Our ablation results are tabulated in Table 3 with the following cases: (a) Initially, we utilize single-shot prompting. With the OD outputs and the high-level textual goal, the LLM independently creates the step-by-step plan in a single-shot (prompt → plan) [4]. (b) Secondly, we incorporate a "memory" mechanism. By appending past interactions to the prompt, we introduce a conversational-style prompting strategy. Here, the LLM outputs only the subsequent textual sub-goal, providing a "pseudo-closed-loop" of textual feedback. However, it lacks true environment grounding since intermediate state cues are inferred only from the conversation history [27]. (c) Finally, we implement grounded feedback via our visual sub-goal generator. As outlined in Section 3, the OD results on these visual sub-goals are added to the prompt for successive prompts.

Our results show a noticeable improvement in task accuracy as we integrate our contributions, reaffirming that our method offers a significant upgrade over existing strategies.

6 Limitations and Future Works

A central drawback of Plan Diffuser is the intertwined nature of its two core components (the LLM planner and diffusion model) which work in an auto-regressive fashion. Should one component falter or misinterpret a step, it can potentially lead to a cascade of subsequent errors, derailing the entire planning process [9]. Furthermore, the generated sub-goals, despite providing guidance to the LLM planner, don't inherently align with the capabilities of the robot arm. Such misalignment might result in synthetic sub-goals that are theoretically sound but infeasible for the robot. Drawing inspirations from SayCan [5], which uses value functions to evaluate the feasibility of skills, extensions to the current work could also consult the robot controllers in the planning loop.

This work also motivates two broader research directions. The first is the development of stronger planning capabilities in robotics by utilizing LLMs equipped with advanced tools, as demonstrated in recent studies by [28, 29]. By connecting these models to code execution interpreters, such as the one proposed by [30], the formality of language in the language-vision interaction can be enhanced, thereby reducing ambiguity in the field. The second general direction is on scaling up the interaction between humans and robots. This includes exploring more modalities beyond text-based prompts, which could lead to more intuitive and effective communication. For example, a combination of language, 2D/3D/4D vision, and haptics could be utilized to convey instructions to robots. Alternatively, when there is ambiguity in human instructions or gestures, a stronger stack of perception and cognitive reasoning is needed using the common-sense capabilities of LLMs.

7 Conclusions

We present Plan Diffuser, a "closed-loop" pipeline for step-by-step planning where each step is augmented with visual feedback of the environment without actual execution. Our method outperforms state-of-the-art methods on 2 language-conditioned tasks in the Ravens benchmark suites, particularly in long-horizon task planning. Furthermore, our approach enjoys robust generalization capabilities in out-of-distribution scenarios, including unseen colors, objects, and increased task complexity. We believe Plan Diffuser represents a significant step towards ensuring LLM-based planning for Embodied AI towards better grounding, generalizability, and safety.

References

[1] Q. M. Marwan, S. C. Chua, and L. C. Kwek. Comprehensive review on reaching and grasping of objects in robotics. *Robotica*, 39(10):1849–1882, 2021.

[2] M. Ersen, E. Oztop, and S. Sariel. Cognition-enabled robot manipulation in human environments: requirements, recent work, and open problems. *IEEE Robotics & Automation Magazine*, 24(3):108–122, 2017.

[3] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[4] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor. Chatgpt for robotics: Design principles and model abilities. *2023*, 2023.

[5] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[6] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591.

[7] R. Wang, J. Mao, J. Hsu, H. Zhao, J. Wu, and Y. Gao. Programmatically grounded, compositionally generalizable robotic manipulation. *arXiv preprint arXiv:2304.13826*, 2023.

[8] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747. PMLR, 2021.

[9] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*, 2022.

[10] Y. Cui, S. Karamcheti, R. Palleti, N. Shivakumar, P. Liang, and D. Sadigh. " no, to the right"–online language corrections for robotic manipulation via shared autonomy. *arXiv preprint arXiv:2301.02555*, 2023.

[11] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser. Tidybot: Personalized robot assistance with large language models. *Autonomous Robots*, 2023.

[12] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530, 2023. doi:10.1109/ICRA48891.2023.10161317.

[13] A. Zhan, R. Zhao, L. Pinto, P. Abbeel, and M. Laskin. Learning visual robotic control efficiently with contrastive pre-training and data augmentation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4040–4047, 2022. doi: 10.1109/IROS47612.2022.9981055.

[14] N. Hansen, Y. Lin, H. Su, X. Wang, V. Kumar, and A. Rajeswaran. Modem: Accelerating visual model-based reinforcement learning with demonstrations. *arXiv preprint*, 2022.

[15] 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*, pages 8748–8763. PMLR, 2021.

[16] S. Huang, Z. Jiang, H. Dong, Y. Qiao, P. Gao, and H. Li. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *arXiv preprint arXiv:2305.11176*, 2023.

[17] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. B. Girshick. Segment anything. *ArXiv*, abs/2304.02643, 2023. URL https://api.semanticscholar.org/CorpusID:257952310.

[18] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. H. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. R. Florence. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:257364842.

[19] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[20] X. Zhao, M. Li, C. Weber, M. B. Hafez, and S. Wermter. Chat with the environment: Interactive multimodal perception using large language models. *arXiv preprint arXiv:2303.08268*, 2023.

[21] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[22] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[23] H. Chase. LangChain. https://github.com/hwchase17/langchain, 2022.

[24] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[25] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[26] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Open-vocabulary object detection via vision and language knowledge distillation. *arXiv preprint arXiv:2104.13921*, 2021.

[27] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg. Text2motion: From natural language instructions to feasible plans. *arXiv preprint arXiv:2303.12153*, 2023.

[28] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*, 2023.

[29] Y. Liang, C. Wu, T. Song, W. Wu, Y. Xia, Y. Liu, Y. Ou, S. Lu, L. Ji, S. Mao, et al. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*, 2023.

[30] D. Surís, S. Menon, and C. Vondrick. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*, 2023.

A Prompts for the LLM

Setup Prompt $\mathcal{P}_{sys}$

```
You are the brain of a pick-place robot. Your task is to generate pick-place commands for the
robot to execute. At each step you will be given the description of the scene and the goal.
The scene description contains information about the object id, object type and bounding box
location coordinates in the format of (x1, y1, x2, y2).
You can use the following commands to control the robot:
- pick ⟨object id⟩ ⟨object type⟩ ⟨pick location⟩
- place ⟨object id⟩ ⟨object type⟩ ⟨place location⟩
You need to generate only one pair of pick-place commands at a time which progresses towards goal
completion. First think step-by-step how to achieve the goal and then generate the commands. If
the goal is achieved, mention "done: goal achieved" and stop generating commands.
Sometimes the scene description might not have the exact color as the goal. In such cases, you
can use the closest colored block to achieve the goal. For example, if the goal is to put a red
block in a green bowl, but the scene description has a pink block instead of a red block, you can
use the pink block to achieve the goal.
A bowl can only hold one block at a time. If you have to move two blocks into two bowls, you need
to place each block in a separate bowl.
A bowl is said to hold a block if the bounding box coordinates of the block is completely inside
the bounding box coordinates of the bowl.
```

Example Conversation $\mathcal{P}_{eg}$ for few-shot prompting

```
# USER
Goal: put the cyan blocks in a green bowl Scene Description:
1. blue block at (121.34, 66.3, 126.29, 72.57)
2. blue bowl at (94.97, 51.14, 131.35, 88.33)
3. cyan bowl at (119.51, 76.78, 122.85, 80.23)
4. cyan block at (14.14, 24.98, 33.2, 44.27)
5. gray block at (58.57, 254.47, 77.5, 272.74)
6. green bowl at (56.94, 211.65, 94.32, 249.07)
7. pink bowl at (21.22, 66.39, 59.18, 103.83)
8. purple block at (60.83, 29.96, 76.93, 45.73)
9. red block at (60.89, 174.48, 78.71, 191.25)
10. yellow bowl at (103.03, 265.14, 140.5, 302.94)

# ASSISTANT
reasoning: there is one cyan block and one green bowl. the cyan block is outside the green bowl.
therefore it has to be moved into the green bowl as per the goal.
pick: 4. cyan block: (14.14, 24.98, 33.2, 44.27)
place: 6. green bowl: (56.94, 211.65, 94.32, 249.07)

# USER
Goal: put the cyan blocks in a green bowl
Scene Description:
1. blue block at (121.34, 66.3, 126.29, 72.57)
2. blue bowl at (94.97, 51.14, 131.35, 88.33)
3. cyan bowl at (119.51, 76.78, 122.85, 80.23)
4. cyan block at (66.94, 221.65, 84.32, 239.07)
5. gray block at (58.57, 254.47, 77.5, 272.74)
6. green bowl at (56.94, 211.65, 94.32, 249.07)
7. pink bowl at (21.22, 66.39, 59.18, 103.83)
8. purple block at (60.83, 29.96, 76.93, 45.73)
9. red block at (60.89, 174.48, 78.71, 191.25)
10. yellow bowl at (103.03, 265.14, 140.5, 302.94)

# ASSISTANT
reasoning: there is one cyan block (id: 4) and one green bowl (id: 6). the cyan block (id: 4)
is inside the green bowl (id: 6). therefore the goal has already been achieved.
done: goal achieved
```