Making VLMs More Robot-Friendly: Self-Critical Distillation of Low-Level Procedural Reasoning

Chan Young Park[*1] **Jillian Fisher**[*1] **Marius Memmel**[1] **Dipika Khullar**[2]
Seoho Yun[1] **Abhishek Gupta**[1] **Yejin Choi**[3]

[1]University of Washington [2]Independent Researcher [3]Stanford University
chanpark@cs.washington.edu jrfish@uw.edu
https://github.com/chan0park/SelfReVision

Abstract

Large language models (LLMs) have shown promise in robotic procedural planning, yet their human-centric reasoning often omits the low-level, grounded details needed for robotic execution. Vision-language models (VLMs) offer a path toward more perceptually grounded plans, but current methods either rely on expensive, large-scale models or are constrained to narrow simulation settings. We introduce SelfReVision, a lightweight and scalable self-improvement framework for vision-language procedural planning. SelfReVision enables small VLMs to iteratively critique, revise, and verify their own plans, without external supervision or teacher models, drawing inspiration from chain-of-thought prompting and self-instruct paradigms. Through this self-distillation loop, models generate higher-quality, execution-ready plans that can be used both at inference and for continued fine-tuning. Using models varying from 3B to 72B, our results show that SelfReVision not only boosts performance over weak base VLMs but also outperforms models 100X the size, yielding improved control in downstream embodied tasks.

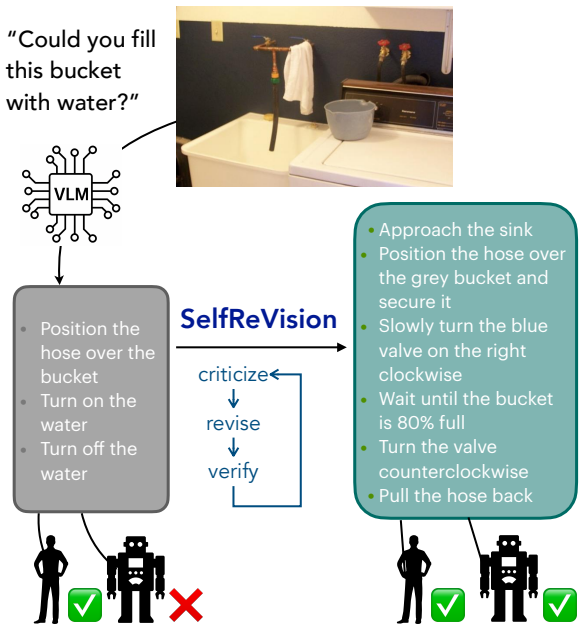


Figure 1: **Overview of SelfReVision**. VLMs tend to generate human-readable plans that are not detailed enough for robotic execution. SelfReVision employs an iterative self-critique, revision, and verification process, to transform initial plans into actionable steps.

1 Introduction

Large language models (LLMs) have recently gained traction as a source of background knowledge for robotic applications, particularly in procedural planning tasks (Huang et al., 2024; Ahn et al., 2022; Shi et al., 2025; Brahman et al., 2023). Their broad pretraining and strong instruction-following capabilities make them appealing tools for generating step-by-step action sequences that, from a human perspective, appear sensible and coherent. Yet, a fundamental challenge remains: because LLMs are trained with human language and human preferences, they tend to generate plans in a way that is intuitive and meaningful to humans, rather than encoding the precise sensory or perceptual details that a robot would need to execute them. As a result, their plans often omit low-level, spatially grounded details essential for execution in the physical world. Consequently, when these plans are applied to robots, they may lead to uncertainty or mistakes in downstream tasks.

Bridging this gap calls for vision-language models (VLMs) that can reason over visual inputs to generate low-level procedural reasoning plans. Yet, current approaches face two critical shortcomings: they either (1) rely on overly specialized setups in simulation environments with limited real-world applicability (Shi et al., 2025), or (2) depend on massive, high-capacity models that are expensive to train and impractical to deploy in many real-world settings (Cheng et al., 2025; OpenAI et al.,

[*]Equal contribution.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 32590–32611
November 4-9, 2025

2024; Yang et al., 2024b). In contrast, many use cases, like in education, robotics, and resource-constrained environments, require solutions that are lightweight, data-efficient, and robust without relying on massive compute. *We argue that strong vision planning can emerge even in smaller VLMs, if they are trained with the right inductive biases and self-improvement strategies.*

We present SelfReVision, a self-improvement framework for vision-language procedural planning based on iterative self-critiquing and self-refining. We show that this method enables small VLMs ranging as small as 3B to 72B, to enhance their performance through self-distillation, *without any external supervision or teacher models*. Inspired by chain-of-thought reasoning and self-instruct methods, we break the task into a three-stage loop: the model first generates an initial plan from a prompt and image, then self-*critiques* it with minimal guidance, self-*revises* the plan accordingly, and finally selects the better of the two via a self-*verification* step. This cycle repeats until the model produces a plan it deems better. The final plans, generated entirely by the model, can be used directly at inference or as self-supervised data to further fine-tune the model and reinforce improvements.

While self-critiquing and self-distillation has been explored in the LLM space (Madaan et al., 2023; Gou et al., 2024), its application to vision-language planning is largely underexplored. To our knowledge, SelfReVision is the first to adapt this paradigm for procedural planning with VLMs. Notably, we apply this method using small, weak base models to emphasize its potential as a tool for enhancing the capabilities of lightweight systems. In addition, we provide a comprehensive ablation study of SelfReVision, showing insights into the role of each component and demonstrating why the iterative loop contributes to performance gains.

To rigorously assess our approach, we introduce a new vision-language evaluation dataset blending real-world and simulation-based visual procedural tasks, an underexplored combination in prior work. We demonstrate our improved VLM plans not only outperform their original base versions, but also surpass state-of-the-art VLMs of 100X larger size. Finally, we show that these enhanced procedural plans translate into better control and execution in downstream embodied agent tasks.

2 Related Works

Procedural Planning LLMs have become increasingly attractive for complex procedural planning tasks (Huang et al., 2024). Pretrained, off-the-shelf LLMs have shown strong performance in this area (Huang et al.; Ahn et al., 2022), and Brahman et al. (2023) further demonstrate that task-specific finetuning can boost their effectiveness even more. Beyond finetuning, another approach used to achieve procedural planning in LLMs is prompting pretrained models to interleave reasoning and action, improving adaptability and decision-making (Huang et al., 2022; Yao et al.). Lastly, some methods instead aim to leverage LLMs for low-level action execution directly, bypassing high-level planning. For example, the Code-as-Policy framework prompts LLMs to produce structured, code-like plans that can be directly interpreted and executed as action sequences (Liang et al., 2023a).

Although LLMs have shown promising results in procedural planning, incorporating vision content can further broaden their practical utility and impact (Ma et al., 2025; Lu et al., 2023). One approach to incorporating visual content is to adopt modular architectures, using specialized encoders to integrate multimodal information from different models (Ilaslan et al., 2024; Kalithasan et al., 2022; Li et al., 2024a; Song et al., 2023; Yang et al., 2023; Zhu et al., 2023). Others enhance performance through finetuning (Driess et al., 2023; Shi et al., 2025) or by optimizing the prompts used with off-the-shelf models (Chen et al., 2024). However, these systems are either large and resource-intensive (Driess et al., 2023) or rely on training data derived from even larger models (Shi et al., 2025).

Self-Distillation and Self-Refinement With the advancement of vision and language models, research has explored using larger, more capable models to generate training data for fine-tuning smaller models, a process commonly referred to as knowledge distillation (Moslemi et al., 2024; Liu et al., 2023; Xu et al., 2024). More recently, however, attention has turned toward self-distillation, in which a weaker model is used to improve itself without relying on a stronger teacher model.

A prominent form of self-distillation involves training data augmentation, where the model generates additional data to further fine-tune itself. This approach has yielded promising results across various domains, including instruction tuning (Wang

et al., 2023), preference modeling (Yang et al., 2024a), and value alignment (Sun et al., 2023). Beyond simply increasing the quantity of data, several studies have demonstrated that filtering the self-generated data can significantly enhance quality. Effective filtering strategies include promoting diversity (Wang et al., 2023), selecting samples based on quality metrics (Jung et al., 2024), and applying external scoring functions to encourage alignment with human values (Gulcehre et al., 2023).

In addition to data generation and filtering, recent work has begun to explore ways in which models can analyze and guide themselves. For instance, some methods use interactive, chain-of-thought-style feedback to help weaker models arrive at correct answers for objective tasks such as math problems and question-answering tasks (Huang et al., 2023; Yu et al., 2024). Similarly, Zheng et al. (2023) employed an LLM-as-Judge approach, using the weaker model itself as a reward model to learn stronger outputs on chatbot tasks. Lastly, similar to our approach, self-feedback and self-refinement techniques have shown promise for LLM tasks such as reasoning (Xie et al., 2023), dialogue response (Madaan et al., 2023), and mathematics (Gou et al., 2024; Madaan et al., 2023). However, these techniques have so far been primarily limited to objective tasks or use outside tools for critiquing (Gou et al., 2024; Xi et al., 2024).

Self-distillation through self-refinement has been explored less frequently in the context of multimodal models, but there are notable exceptions, particularly in image captioning. For example, Wu et al. (2025) proposed a method where the model generates intermediate reasoning hidden states, which are then used to retrain the base model, effectively improving performance through internal feedback. Other studies have leveraged self-distillation to augment human-annotated datasets, enriching the training corpus with additional synthetic examples (Deng et al., 2024; Fang et al., 2024). These approaches suggest that even in multimodal settings, self-distillation can provide valuable improvements when carefully designed.

3 Methods

Procedural planning involves generating a step-by-step plan to achieve a goal. We focus on open-ended, multi-step tasks with diverse, valid solutions. Unlike prior work relying on powerful LLMs in purely textual settings, we tackle a harder, more realistic problem: vision-grounded procedural planning using only weak VLMs. This multimodal setup adds complexity, plans must align with user intent and visual constraints like spatial layout, semantics, and object presence. We further restrict ourselves to low-capacity models, reflecting deployment in resource-limited settings without large teacher models or gold labels. To meet this challenge, we propose a self-distillation framework where a weak model improves through its own reasoning, via a structured loop of critique, revision, and verification, without external supervision or extra data.

Algorithm 1 SelfReVision

Input: Model θ, Image x, Instruction I,
  Generate initial plan: $p_0 \leftarrow \theta(x, I)$
  Initialize: $p_{\text{curr}} \leftarrow p_0$
  repeat
    Critique: $c \leftarrow \text{Crit}(p_{\text{curr}})$ // Self-critique
    Revise: $p_{\text{rev}} \leftarrow \text{Rev}(p_{\text{curr}}, c)$ // Generate improved plan
    Verify: $p_{\text{best}} \leftarrow \text{Ver}(p_{\text{curr}}, p_{\text{rev}})$ // Choose better plan
    if $p_{\text{best}} = p_{\text{rev}}$ **then**
      break // Improvement found; terminate loop
    else
      continue // No improvement; keep current plan and revise again
    end if
  until Convergence or max iterations reached
  return Final plan p_{curr}

Self-Distillation via Self-Improvement We build on the principle of self-distillation, a training paradigm where a model improves itself by learning from its own outputs. Unlike classical knowledge distillation, which requires a stronger teacher model, our approach is entirely self-supervised. Let θ denote a base model. We define a self-distilled dataset D as:

$$D = (x, y, \phi_{\text{sd}}(x, y)) \mid x \sim \mathcal{X},\ y \sim p_\theta(y \mid x, I),$$

where x is an input prompt, I is an instruction or task description, and y is the model's own initial plan output. The transformation function ϕ_{sd} refines this output via a structured process involving targeted critique and revision.

SelfReVision We introduce SelfReVision, a three-stage Criticize–Revise–Verify pipeline to instantiate ϕ_{sd}. This process encourages the model

		Places							Simulation						
		Coverage	Ordering	Complete	Image.	Overall	Imp.↑	+#Inf	Coverage	Ordering	Complete	Image.	Overall	Imp.↑	+#Inf
Qwen-3B	⇔ GPT-4o	1 ⇔ 95	6 ⇔ 83	1 ⇔ 97	3 ⇔ 60	0 ⇔ 97	97	0	4 ⇔ 95	7 ⇔ 84	2 ⇔ 98	5 ⇔ 50	1 ⇔ 98	97	0
	⇔ PaliGemma	91 ⇔ 8	89 ⇔ 5	90 ⇔ 8	82 ⇔ 4	92 ⇔ 7	-85	0	95 ⇔ 4	88 ⇔ 7	93 ⇔ 7	85 ⇔ 3	95 ⇔ 4	-91	0
	⇔ Basic Distillation	27 ⇔ 64	33 ⇔ 53	26 ⇔ 72	25 ⇔ 30	27 ⇔ 69	42	0	25 ⇔ 64	34 ⇔ 44	28 ⇔ 72	20 ⇔ 15	28 ⇔ 66	38	0
	⇔ Best-of-N	31 ⇔ 44	32 ⇔ 41	41 ⇔ 54	18 ⇔ 28	38 ⇔ 58	20	6	46 ⇔ 35	43 ⇔ 23	53 ⇔ 40	16 ⇔ 21	51 ⇔ 40	-11	6
	⇔ SelfReVision	9 ⇔ 52	17 ⇔ 28	6 ⇔ 62	9 ⇔ 12	15 ⇔ 61	46	9.1	6 ⇔ 52	20 ⇔ 26	6 ⇔ 74	11 ⇔ 18	12 ⇔ 67	55	7.2
	⇔ SelfReVision+SFT	25 ⇔ 59	25 ⇔ 49	29 ⇔ 68	25 ⇔ 26	30 ⇔ 68	38	0	29 ⇔ 58	30 ⇔ 37	34 ⇔ 63	17 ⇔ 25	32 ⇔ 61	29	0
Gemma-4B	⇔ GPT-4o	8 ⇔ 80	15 ⇔ 69	12 ⇔ 81	13 ⇔ 54	11 ⇔ 89	78	0	15 ⇔ 59	28 ⇔ 53	31 ⇔ 68	22 ⇔ 35	25 ⇔ 73	48	0
	⇔ PaliGemma	97 ⇔ 3	92 ⇔ 4	100 ⇔ 0	87 ⇔ 4	97 ⇔ 3	-94	0	98 ⇔ 2	95 ⇔ 3	98 ⇔ 1	91 ⇔ 1	98 ⇔ 2	-96	0
	⇔ Basic Distillation	64 ⇔ 22	69 ⇔ 19	75 ⇔ 24	53 ⇔ 18	76 ⇔ 22	-54	0	71 ⇔ 18	72 ⇔ 17	81 ⇔ 14	47 ⇔ 13	82 ⇔ 18	-64	0
	⇔ Best-of-N	26 ⇔ 44	30 ⇔ 35	30 ⇔ 54	18 ⇔ 32	33 ⇔ 57	24	6	28 ⇔ 39	29 ⇔ 30	40 ⇔ 51	14 ⇔ 27	39 ⇔ 52	13	6
	⇔ SelfReVision	8 ⇔ 73	26 ⇔ 53	6 ⇔ 88	16 ⇔ 42	8 ⇔ 86	78	8.9	10 ⇔ 73	33 ⇔ 53	5 ⇔ 90	17 ⇔ 25	13 ⇔ 82	69	7.0
	⇔ SelfReVision+SFT	32 ⇔ 56	33 ⇔ 57	39 ⇔ 58	27 ⇔ 41	36 ⇔ 60	24	0	33 ⇔ 45	35 ⇔ 47	38 ⇔ 56	28 ⇔ 45	36 ⇔ 54	11	0
Qwen-7B	⇔ GPT-4o	11 ⇔ 76	19 ⇔ 66	11 ⇔ 85	10 ⇔ 48	10 ⇔ 86	76	0	11 ⇔ 76	18 ⇔ 66	10 ⇔ 87	10 ⇔ 44	10 ⇔ 86	76	0
	⇔ PaliGemma	99 ⇔ 0	95 ⇔ 2	100 ⇔ 0	82 ⇔ 3	99 ⇔ 1	-98	0	98 ⇔ 2	96 ⇔ 1	99 ⇔ 1	87 ⇔ 1	98 ⇔ 2	-96	0
	⇔ Basic Distillation	43 ⇔ 35	36 ⇔ 33	53 ⇔ 42	27 ⇔ 20	54 ⇔ 37	-17	0	43 ⇔ 31	38 ⇔ 31	47 ⇔ 46	23 ⇔ 14	51 ⇔ 44	-7	0
	⇔ Best-of-N	29 ⇔ 48	31 ⇔ 43	42 ⇔ 51	12 ⇔ 36	38 ⇔ 58	20	6	35 ⇔ 41	32 ⇔ 25	43 ⇔ 47	15 ⇔ 25	41 ⇔ 49	8	6
	⇔ SelfReVision	3 ⇔ 71	30 ⇔ 31	3 ⇔ 91	17 ⇔ 38	9 ⇔ 82	73	9.2	2 ⇔ 75	38 ⇔ 21	5 ⇔ 89	12 ⇔ 16	7 ⇔ 86	79	10.0
	⇔ SelfReVision+SFT	20 ⇔ 64	35 ⇔ 46	20 ⇔ 77	22 ⇔ 43	18 ⇔ 75	57	0	17 ⇔ 75	37 ⇔ 44	19 ⇔ 79	19 ⇔ 22	21 ⇔ 75	54	0
Gemma-12B	⇔ GPT-4o	24 ⇔ 49	22 ⇔ 54	37 ⇔ 56	25 ⇔ 31	32 ⇔ 56	24	0	38 ⇔ 41	32 ⇔ 52	44 ⇔ 54	22 ⇔ 32	44 ⇔ 55	11	0
	⇔ PaliGemma	100 ⇔ 0	97 ⇔ 1	100 ⇔ 0	90 ⇔ 0	100 ⇔ 0	-100	0	100 ⇔ 0	100 ⇔ 0	100 ⇔ 0	91 ⇔ 1	100 ⇔ 0	-100	0
	⇔ Basic Distillation	78 ⇔ 12	66 ⇔ 17	85 ⇔ 12	44 ⇔ 15	87 ⇔ 12	-75	0	70 ⇔ 11	66 ⇔ 13	86 ⇔ 12	53 ⇔ 8	89 ⇔ 9	-80	0
	⇔ Best-of-N	23 ⇔ 40	31 ⇔ 33	33 ⇔ 55	19 ⇔ 33	29 ⇔ 54	25	6	18 ⇔ 41	21 ⇔ 46	29 ⇔ 55	8 ⇔ 26	28 ⇔ 61	33	6
	⇔ SelfReVision	8 ⇔ 79	51 ⇔ 31	6 ⇔ 91	36 ⇔ 35	10 ⇔ 80	70	6.7	8 ⇔ 84	50 ⇔ 35	6 ⇔ 93	29 ⇔ 19	11 ⇔ 81	70	6.6
	⇔ SelfReVision+SFT	24 ⇔ 64	43 ⇔ 45	22 ⇔ 77	38 ⇔ 26	23 ⇔ 72	49	0	16 ⇔ 71	49 ⇔ 32	17 ⇔ 81	32 ⇔ 22	24 ⇔ 70	46	0
Gemma-27B	⇔ GPT-4o	31 ⇔ 45	29 ⇔ 50	42 ⇔ 53	31 ⇔ 31	39 ⇔ 53	14	0	38 ⇔ 34	30 ⇔ 46	46 ⇔ 47	34 ⇔ 17	46 ⇔ 48	2	0
	⇔ PaliGemma	100 ⇔ 0	98 ⇔ 2	100 ⇔ 0	98 ⇔ 2	100 ⇔ 0	-100	0	99 ⇔ 1	96 ⇔ 2	99 ⇔ 1	95 ⇔ 1	99 ⇔ 1	-98	0
	⇔ Basic Distillation	82 ⇔ 8	55 ⇔ 22	88 ⇔ 9	48 ⇔ 8	86 ⇔ 10	-76	0	74 ⇔ 8	58 ⇔ 19	87 ⇔ 11	48 ⇔ 10	86 ⇔ 11	-75	0
	⇔ Best-of-N	22 ⇔ 37	26 ⇔ 28	34 ⇔ 55	17 ⇔ 24	35 ⇔ 53	18	6	23 ⇔ 38	27 ⇔ 28	37 ⇔ 54	15 ⇔ 26	31 ⇔ 58	27	6
	⇔ SelfReVision	6 ⇔ 85	50 ⇔ 34	1 ⇔ 97	28 ⇔ 21	7 ⇔ 86	79	6.6	4 ⇔ 89	39 ⇔ 48	3 ⇔ 97	36 ⇔ 22	7 ⇔ 88	81	6.2
Qwen-32B	⇔ GPT-4o	51 ⇔ 20	28 ⇔ 40	74 ⇔ 20	26 ⇔ 29	63 ⇔ 32	-31	0	52 ⇔ 19	31 ⇔ 44	74 ⇔ 17	25 ⇔ 27	71 ⇔ 26	-45	0
	⇔ PaliGemma	100 ⇔ 0	99 ⇔ 0	100 ⇔ 0	90 ⇔ 2	100 ⇔ 0	-100	0	100 ⇔ 0	98 ⇔ 1	100 ⇔ 0	90 ⇔ 0	100 ⇔ 0	-100	0
	⇔ Basic Distillation	68 ⇔ 13	50 ⇔ 20	77 ⇔ 19	36 ⇔ 13	81 ⇔ 13	-68	0	61 ⇔ 13	51 ⇔ 24	72 ⇔ 21	34 ⇔ 13	74 ⇔ 22	-52	0
	⇔ Best-of-N	22 ⇔ 43	27 ⇔ 29	31 ⇔ 65	21 ⇔ 25	37 ⇔ 54	17	6	18 ⇔ 52	21 ⇔ 41	24 ⇔ 65	17 ⇔ 17	27 ⇔ 65	38	6
	⇔ SelfReVision	2 ⇔ 60	28 ⇔ 23	1 ⇔ 62	21 ⇔ 9	3 ⇔ 56	53	16	0 ⇔ 69	28 ⇔ 31	0 ⇔ 78	15 ⇔ 14	4 ⇔ 72	68	12.7
Qwen-72B	⇔ GPT-4o	14 ⇔ 61	20 ⇔ 59	17 ⇔ 76	17 ⇔ 35	15 ⇔ 76	61	0	11 ⇔ 62	15 ⇔ 70	19 ⇔ 79	16 ⇔ 43	13 ⇔ 82	69	0
	⇔ PaliGemma	98 ⇔ 1	95 ⇔ 4	100 ⇔ 0	88 ⇔ 4	98 ⇔ 1	97	0	99 ⇔ 1	98 ⇔ 1	100 ⇔ 0	94 ⇔ 0	99 ⇔ 1	-98	0
	⇔ Basic Distillation	66 ⇔ 14	53 ⇔ 28	76 ⇔ 21	35 ⇔ 22	73 ⇔ 22	-51	0	48 ⇔ 30	39 ⇔ 38	59 ⇔ 34	25 ⇔ 22	59 ⇔ 37	-22	0
	⇔ Best-of-N	23 ⇔ 56	23 ⇔ 52	25 ⇔ 72	15 ⇔ 37	25 ⇔ 72	47	6	12 ⇔ 60	21 ⇔ 40	11 ⇔ 85	14 ⇔ 27	13 ⇔ 81	68	6
	⇔ SelfReVision	4 ⇔ 89	49 ⇔ 39	2 ⇔ 98	21 ⇔ 38	6 ⇔ 85	79	7.3	5 ⇔ 90	31 ⇔ 53	2 ⇔ 97	25 ⇔ 26	8 ⇔ 89	81	6.5

Table 1: **Win rate comparison of baseline models and SelfReVision against initial plan** p_0, across two datasets (PLACES and SIMULATION). Evaluation is done using GPT4o as judge across five dimensions, with overall improvement (Imp.) showing the win rate difference and higher values indicating better plan quality. The +#Inf column shows the average number of inference calls required by each method (0 for single-pass methods, 6 for Best-of-N with N=5, and 7-16 for SelfReVision depending on when the verification step accepts an improved plan).

to iteratively refine its outputs via structured introspection:

- Criticize (Crit): The model generates an initial plan $p_0 = \theta(x, I)$, which may be vague, image-agnostic, or incomplete. We then prompt the model to produce a critical self-assessment $\text{Crit}(p_0)$.
- Revise (Rev): Using its self-generated critique, the model produces a revised plan $p_1 = \text{Rev}(p_0, \text{Crit}(p_0))$. This phase encourages localized, meaningful improvements, including action-grounded comments, splitting complex revisions into manageable subgoals via chain-of-thought prompting.
- Verify (Ver): Finally, the model evaluates both p_0 and p_1 to decide which is superior: $p_{\text{best}} = \text{Ver}(p_0, p_1)$. If the revised plan is preferred, the process terminates, if is not preferred then the process continues recursively until a better plan is produced.

The iterative nature of this loop is formalized in algorithm 1, and can be used to run for any threshold amount of refinement loops (i.e. rounds). It can also be used to generate a set amount of final plans p_{curr} that can then be compared to the baseline or each-other. This closed-loop formulation mimics aspects of human self-improvement, which identifies flaws, attempt revision, and critically evaluate the result.

Inference vs. Finetuning SelfReVision generates curated outputs through self-distillation, which can be leveraged in two ways: used directly at inference time or as training data for finetuning. Using SelfReVision at inference time requires no model updates and allows fast deployment, but may incur computational overhead or complexity in orchestration. In contrast, finetuning incorporates the improvements directly into the model, enabling faster inference and better generalization, but requires additional training time and resources. The choice

"Could you bring me the red poster from the wall?"

"Pack two cookies and one fruit which is high in potassium for a snack."

Figure 2: **Evaluation examples** from the real-world PLACES dataset (Zhou et al., 2017) (right) and from the SIMULATION dataset, VirtualHome (Puig et al., 2018) and BEHAVIOR-100 (Srivastava et al., 2022) (left).

depends on the desired balance between flexibility, performance, and scalability.

4 Experiments

We conduct two types of experiments to evaluate SelfReVision for planning: image-based procedural planning (§4.1) and embodied agent tasks (§4.2). The image-based procedural planning experiments assess the effectiveness of SelfReVision in vision-language planning and provide insights into the types of self-reflection that are helpful for planning. We then evaluate directly on embodied agent tasks to demonstrate how SelfReVision results in direct improvements in vision-language procedural planning for embodied agents.

SelfReVision Implementation Details We used a diverse range of base models to experiment with SelfReVision; Qwen-2.5-VL-Instruct (3B, 7B, 32B, 72B) (Bai et al., 2025) and Gemma 3 (4B, 12B, 27B) (Team et al., 2024). Among open-sourced VLMs, these models have been shown to perform well on visual reasoning tasks (Cheng et al., 2024).

Guided by a scaling experiment with number of revisions per round, we set the number of revisions to 2 for our main experiments. We set the number of maximum rounds to 5. For training, we set the temperature of the critique and refine stage to 0.5, while we use greedy decoding for the initial planning and validation stage.

We implement the SelfReVision as both an inference-time method (SelfReVision) and as supervised-finetuning (SelfReVision+SFT). For the SelfReVi+SFT method we curated a $n = 160K$ subset of images from the PLACES Dataset (Zhou et al., 2017), which contains real-world scenes categorized by location type (e.g., airport lounge, kitchen, barn). We selected a diverse range of both indoor and outdoor scenes. Next, we used GPT-4o (OpenAI et al., 2024) to generate a variety of plausible goals that a user might want to achieve in each given setting. Full experimental details are provided in Appendix A.

4.1 Goal-Based Procedural Planning

Evaluation Dataset We evaluated SelfReVision on both real-world and simulation settings, as both settings frequently require procedural planning. For the real-world setting, we used a held-out test set of $n = 100$ image and user-input pairs sampled from the PLACES Dataset (Zhou et al., 2017), and the corresponding user inputs were generated using GPT-4o (OpenAI et al., 2024).

For the SIMULATION setting, we used a modified version of the MFE-ETP dataset (Zhang et al., 2024), which consists of $n = 100$ image and user-prompt pairs drawn from the popular procedural simulation environments VirtualHome (Puig et al., 2018) and BEHAVIOR-100 (Srivastava et al., 2022). Since this dataset includes multi-image scenarios, we adjusted some user inputs to correspond to a single selected image when necessary. Example inputs and visualizations are shown in Figure 2, with additional details provided in Appendix A.2.

Evaluation Metrics Prior work (Brahman et al., 2023; Huang et al.) has evaluated procedural plans based on four dimensions: Coverage, Ordering, Completeness, and Overall Quality. We extend this framework by introducing a fifth criterion—Image Groundedness—to assess how well a plan aligns with the visual context. Specifically we define these criteria as:

- **Coverage**: How well the plan addresses the user's input.
- **Ordering**: Whether the steps follow a logical and coherent sequence.
- **Completeness**: Whether the plan is sufficiently detailed and informative.
- **Image Groundedness**: Whether the plan is plausible given the visual scene.
- **Overall Quality**: The overall effectiveness and appropriateness of the plan.

Given the strong performance of LLMs-as-judges (Zheng et al., 2023), we use GPT-4o (OpenAI et al., 2024) as an automated evaluator via prompting. To validate this approach, we measured inter-rater reliability on a sample of $n = 60$ and found an average agreement of 0.54 between three GPT-4o judgements and three human annotators.

		Places						Simulation					
		Coverage	Ordering	Complete	Image.	Overall	Imp.↑	Coverage	Ordering	Complete	Image.	Overall	Imp.↑
GPT-4o	⇔ Qwen-3B	91 ⇔ 7	82 ⇔ 9	91 ⇔ 7	56 ⇔ 6	92 ⇔ 3	-89	83 ⇔ 9	82 ⇔ 5	87 ⇔ 13	52 ⇔ 9	93 ⇔ 3	-90
	⇔ Gemma-4B	47 ⇔ 32	73 ⇔ 17	56 ⇔ 43	48 ⇔ 18	57 ⇔ 36	-21	44 ⇔37	71 ⇔23	47 ⇔ 48	45 ⇔ 14	58 ⇔ 38	-20
	⇔ Qwen-7B	54 ⇔ 27	69 ⇔ 21	55 ⇔ 42	44 ⇔ 15	60 ⇔ 30	-30	55 ⇔ 30	75 ⇔ 15	61 ⇔ 37	42 ⇔ 15	62 ⇔ 31	-31
	⇔ Gemma-12B	23 ⇔ 67	69 ⇔ 21	16 ⇔ 82	45 ⇔ 25	26 ⇔ 65	39	17 ⇔ 74	73 ⇔ 22	15 ⇔ 84	39 ⇔ 20	25 ⇔ 68	43
	⇔ Gemma-27B	15 ⇔ 73	46 ⇔ 40	11 ⇔ 88	38 ⇔ 24	20 ⇔ 70	50	10 ⇔ 82	41 ⇔ 42	7 ⇔ 92	34 ⇔ 19	9 ⇔ 81	72
	⇔ Qwen-32B	11 ⇔ 76	53 ⇔ 32	9 ⇔ 91	34 ⇔ 20	15 ⇔ 82	67	5 ⇔ 83	44 ⇔ 36	7 ⇔ 91	38 ⇔ 12	10 ⇔ 84	74
	⇔ Qwen-72B	27 ⇔ 63	72 ⇔ 18	19 ⇔ 78	42 ⇔ 33	32 ⇔ 58	26	19 ⇔ 71	65 ⇔ 25	19 ⇔ 81	32 ⇔ 17	25 ⇔ 69	44

Table 2: **Win rate comparison of GPT4o and SelfReVision plans directly**, across two datasets (PLACES and SIMULATION). Evaluation is done using GPT4o as judge across five dimensions, including overall improvement (Imp.). Higher improvement indicates better plan quality.

		Coverage	Ordering	Complete	Image	Overall	Imp.↑
Places	CRV	5.7 ⇔ 72.7	35.9 ⇔ 34.1	3.6 ⇔ 84.1	21.1 ⇔ 27.9	8.3 ⇔ 76.6	**68.3**
	CR	9.4 ⇔ 67.0	37.4 ⇔ 30.7	7.1 ⇔ 78.1	25.6 ⇔ 24.4	11.3 ⇔ 70.3	**59.0**
	RV	7.4 ⇔ 34.7	14.4 ⇔ 19.4	7.1 ⇔ 56.6	6.6 ⇔ 26.4	13.3 ⇔ 60.0	**46.7**
	R	9.6 ⇔ 32.7	12.9 ⇔ 18.4	10.3 ⇔ 52.1	8.0 ⇔ 24.0	16.9 ⇔ 55.0	**38.1**
Simulation	CRV	5.0 ⇔ 76.0	34.1 ⇔ 38.1	3.9 ⇔ 88.3	20.7 ⇔ 20.0	8.9 ⇔ 80.7	**71.9**
	CR	6.3 ⇔ 71.3	34.4 ⇔ 36.4	4.4 ⇔ 81.7	21.6 ⇔ 20.3	9.6 ⇔ 73.7	**64.1**
	RV	8.6 ⇔ 31.9	15.1 ⇔ 18.3	9.0 ⇔ 54.3	6.3 ⇔ 22.6	16.1 ⇔ 55.4	**39.3**
	R	8.4 ⇔ 31.1	15.3 ⇔ 18.6	9.9 ⇔ 55.1	7.1 ⇔ 23.4	16.1 ⇔ 57.1	**41.0**

Table 3: **Ablation models' win rate comparison** against p_0 across five evaluation dimensions and overall improvement (Imp.).

This level of agreement is in line with the average agreement between humans. See Appendix B for full details.

For our primary evaluation metric, we report the *win rate*, which is the percentage of samples in which the revised plan (or model output) is preferred over that of the base model (i.e. p_0).

Baselines To demonstrate the effectiveness of SelfReVision, we first compare the refined plans to the initial plans generated by the models using few-shot prompting. We also evaluate responses from other baselines such as GPT-4o (representing a powerful large model) (OpenAI et al., 2024), PaliGemma (a domain-specific model trained for planning) (Beyer et al., 2024), Basic Distillation (inspired by Shi et al. (2025), we use a more detailed self-distillation prompting that includes instructions for physical and spatial grounding) and best-of-N (an inference-time algorithm that generates multiple outputs and selects the best one). The prompts and examples provided for GPT-4o and PaliGemma match those given to the base models. For the best-of-N baseline, we use $N=5$: we sample five different plans with a temperature of 0.5, followed by a final inference step to select the best plan among them. This setup approximately matches the number of additional inferences made by both SelfReVision and the baseline. For the prompts used in all methods see Appendix A.

Results: SelfReVision yields large and consistent improvements over baselines. Table 1 shows that across all model sizes and both datasets, SelfReVision consistently outperforms the initial plans p_0 by wide margins. Specifically, there is an average win rate 68% on PLACES and 72% on SIMULATION, with the most dramatic gains in completeness and coverage, often surpassing 80% win rates against the base plans. These results demonstrate that iterative self-improvement through SelfReVision is highly effective in enhancing the structure, richness, and plausibility of plans, regardless of model size. Notably, larger models tend to benefit even more from SelfReVision, both in absolute win rates and in the consistency of gains across metrics. For example, models over 12B have on average 74% gain overall using SelfReVision compared to 68% for models 12B and under.

Compared to alternative methods such as BEST-OF-N sampling, Basic Distillation prompting, and PaliGemma, SelfReVision shows clear superiority. While Best-of-N offers modest improvements for small models (8% − 38%), SelfReVision provides substantially higher gains (60% across most settings). Somewhat unexpectedly, PaliGemma, a strong pretrained VLM, consistently underperforms, losing over 90% of matchups across both datasets. Despite being trained on image distributions similar to those in PLACES, it appears to lack the procedural reasoning abilities required for grounded multi-step planning, suggesting its limitations in this domain. Lastly, we see similar poor performance from the detailed prompting used in Basic Distillation with only moderate gains using the smallest model Qwen-3B. This underscores the need for more principled self-distillation methods like our proposed SelfReVision.

Lastly, we also assess the impact of SFT on SelfReVision outputs. While SelfReVision+SFT achieves moderate gains in some settings (e.g., 57%/54% for Qwen-7B in PLACES and SIMULATION), it often underperforms compared to the inference-time method and in several cases yields no improvement. This performance gap stems from their fundamentally different mecha-

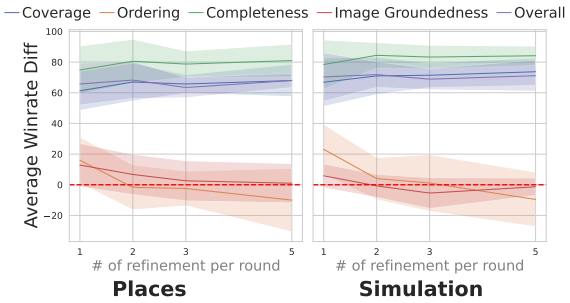


Figure 3: **Average win rate difference** (win rate of SelfReVision - p_0) over number of refinement per round.

nisms: the inference-time SelfReVision leverages the model's full reasoning capacity through iterative self-critique and revision, while SFT trains the model to directly imitate the final refined outputs in a single pass. Our results suggest that this imitation approach cannot fully capture the underlying iterative reasoning process that generated the improvements. The gap is particularly pronounced for larger models (12B+), where SFT may overfit to surface patterns in the refined plans rather than learning the deeper reasoning process. Thus, while SFT offers significant efficiency gains, requiring only one forward pass at inference, this comes at the cost of plan quality for complex tasks that benefit from iterative refinement.

Results: SelfReVision produce better plans than GPT4o. To assess how SelfReVision stacks up against significantly larger models, we compare the win rate of plans it generates with those produced by GPT4o, as shown in Table 2. Our results reveal that for models with 12B parameters or more, SelfReVision achieves a win rate at least 25% higher than GPT-4o. This highlights the effectiveness of self-critical, self-revision strategies in enabling even smaller models to outperform much larger ones.

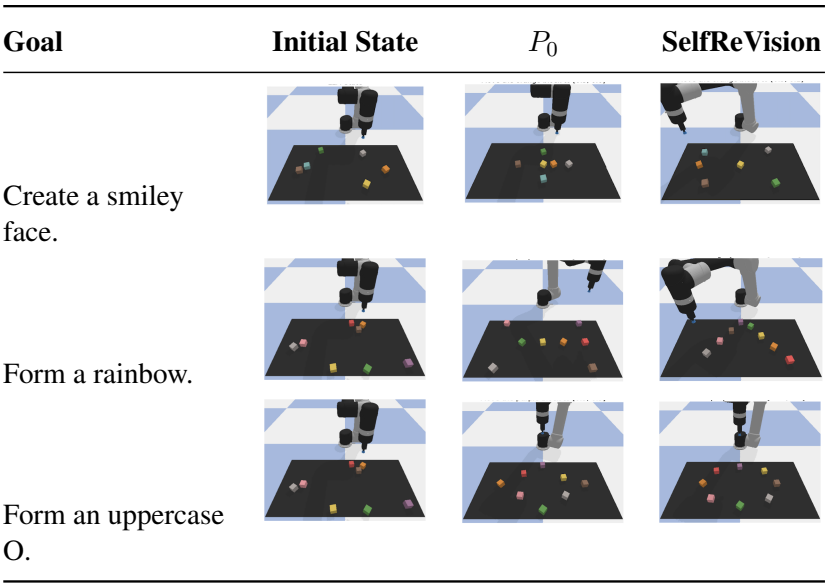
Results: Tradeoffs in Refinement Scaling We examined how SelfReVision's performance changes with more refinement cycles in its self-refinement loop. As shown in Figure 3, the average Overall win rate rises from 75% to 81% on PLACES and from 78% to 81% on SIMULATION as the number of rounds increases from 1 to 5. However, the gains vary by metric: Coverage and Completeness steadily improve (e.g., +11 and +10 on PLACES), suggesting that additional rounds help produce more thorough plans. In contrast, Ordering and Image-Groundedness decline slightly (–5 and –3), indicating that later rounds may introduce speculative or less visually anchored content. Early refinements tend to add useful specifics (e.g., "80% fill"), while later ones often bring more tentative phrasing (e.g., "if there is water in the cup"), reflecting a trade-off between elaboration and precision.

Notably, most of the improvement occurs within the first 2–3 rounds, showing that a few iterations are often enough to achieve strong results without sacrificing clarity. This finding has important practical implications: while the full SelfReVision pipeline averages 8 forward passes at maximum 5 rounds, practitioners can achieve most benefits with just 2-3 rounds, significantly reducing inference overhead. This computational cost remains competitive with API-based alternatives like GPT-4o, which incur high costs and network latency. For deployment scenarios requiring single-pass inference, SelfReVision+SFT provides a viable alternative by distilling these improvements into the model weights.

Results: Ablation of Pipeline Steps To evaluate the contribution of each component in SelfReVision self-refinement loop, we conducted a series of ablation experiments by selectively removing individual stages. Table 3 presents the ablation results on both the PLACES and SIMULATION datasets, averaged across the seven VLMs. We compare four configurations: the full CRV (Criticize-Revise-Verify) pipeline, CR (Criticize-Revise), RV (Revise-Verify), and R (Revise-only). The details on ablation model variants can be found in Appendix A.2.

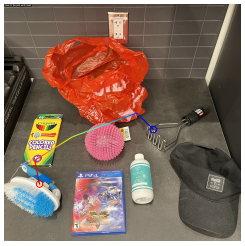
The full CRV pipeline yields the strongest performance, with average win-rate improvements of 68.3% on the PLACES dataset and 71.9% on the SIMULATION dataset. This result confirms that integrating all three stages produces the most robust improvements in procedural plan quality. Notably, compared to CR, we observe significantly larger performance drops with RV and R. These variants especially show reduced improvements in Coverage and Completeness, indicating the essential role of the Criticize step in generating more comprehensive plans that better address user requests.

While the CR variant demonstrates the best performance among the ablated configurations, it still exhibits notable performance drops (-9.3% on PLACES and -7.8% on SIMULATION) relative to the full CRV. In some cases, the refined plans were



(a) Block-building goals, initial state, P_0, and SelfReVision outputs.

Addition: *"place the green tupperware lid on the green tupperware"*



Removal: *"pick up and place the pink scrub brush into the red bag"*

(b) Object manipulation with SelfReVision in hierarchical planning.

Figure 4: **Examples from two embodied agent tasks**: (a) block-building goals, initial setting, and then finalized setting after running P_0, and SelfReVision plans. The first two rows show examples from Gemma 12B and the last row is from Gemma 27B.; (b) examples of correct addition and removal of SelfReVision plan in hierarchal planning.

even worse than the initial plans in terms of Ordering and Image Groundedness. These results suggest that the Verify step plays a critical role in filtering out suboptimal revisions–particularly those that disrupt the correct order or misalign with visual context. Together, these findings underscore that each stage in SelfReVision contributes distinct and complementary benefits to plan refinement.

Qualitative Analysis Figure 5 provides an example of an initial plan generated by Gemma-27B, along with the self-feedback and refined plan produced by SelfReVision. Although the initial plan seems sufficiently clear at first glance, the self-critique step identifies critical shortcomings such as positioning the hose after turning on the water and potential interference by placing the towel too close to the bucket. The refined plan explicitly addresses these issues (e.g. "place the towel away from the bucket," "secure the end of the hose inside the bucket before turning on the water"). Additionally, the refined version includes explicit instructions regarding robot-specific considerations– monitoring for leaks or splashes– details intuitive to humans but essential for robotic execution. This iterative refinement thus results in a more robust and executable plan.

4.2 Application to Embodied Agents

To study the ability of SelfReVision to improve planning in embodied settings, we construct two challenging scenarios: (1) a simulated pick-and-place environment (Zeng et al., 2020) controlled by code-as-policies (Liang et al., 2023b) and (2) a real-world planning environment based on path prediction inspired by HAMSTER (Li et al., 2024b). We limit our evaluation to the models that were the best in baseline procedural planning, Gemma 12B and 27B (Team et al., 2024).

Evaluation Dataset For the simulated pick-and-place environments, we first curated 14 semantically unique manipulation goals (e.g., "Form a shape of an uppercase X with the blocks", "Create a smiley face") and paired them with 8 different initial block configurations involving 6 or 8 blocks from Zeng et al. (2020). This yielded a total of $n = 112$ samples. For the real-world setting, we created 10 real scenarios across three environments – kitchen, workshop, and office – each involving a high-level task (e.g., "Pack items for a children's lunch").

Evaluation Metrics For the simulated pick-and-place environment, we ran each plan using a code-as-policies simulator (Liang et al., 2023b) which generated a static image for each step. Then, a hu-

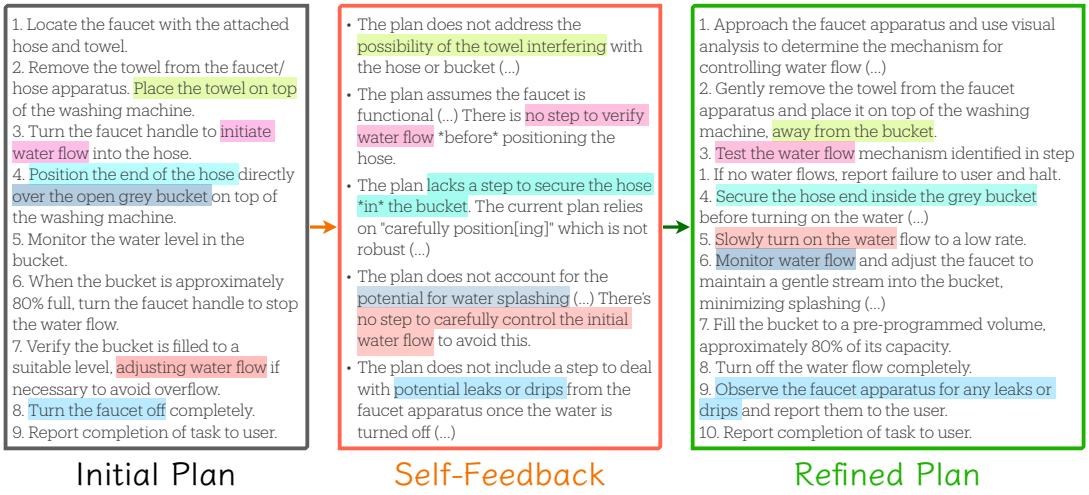


Figure 5: Initial plan, feedback, and refined plan generated by Gemma-27B for the example in Figure 1.

	Gemma-12b		Gemma-27b	
	p_0	SelfReVis.	p_0	SelfReVis.
6 Blocks	0.16	**0.45**	0.36	**0.59**
8 Blocks	0.14	**0.39**	0.29	**0.39**

Table 4: Results of the simulated block manipulation tasks, showing the average success rate for both the baseline p_0 and SelfReVision plans on settings with 6 or 8 blocks over Gemma 12B and 27B.

man rater evaluated the final configuration, judging whether the plan achieved the stated goal. For real world settings, we used Li et al. (2024b) to generate a trace path for each step in each generated plan. Then, a human rater assessed whether each individual step was completed successfully by the generated trace.

Results: SelfReVision improves downstream performance on block manipulation task and real-world planning scenarios As shown in Table 4, the plans enhanced by SelfReVision outperformed the base model plans by 26% (12B) and 17% (27B), respectively. Qualitatively, the improvements were especially notable in more complex tasks like "Create a smiley face" or "Form a rainbow". For the smaller 12B model, SelfReVision often transformed failed attempts into successful plans (see Figure 4a). In contrast, for the larger 27B model, the improvements were more subtle, enhancing already successful outputs, such as making the structure more rounded in the final example of Figure 4a. These results indicate that the critical revision process introduced by SelfReVision can produce higher-quality plans that more reliably complete manipulation tasks.

For the hierarchical task, we found that the SelfReVision plans resulted in 70% successful traces creation by the HAMSTER action model compared to only 61% of the base model plans. These improvements stemmed from both meaningful additions and removals within the plans, resulting in more accurate downstream traces. Figure 4b presents two illustrative examples of such revisions and their downstream impact. In the top image, where the goal was to pack a kid's lunch, SelfReVision correctly added a final missing step to place the lid on the Tupperware. In contrast, the bottom image shows an error in the base plan for the goal "pack toys for a kid," where the model mistakenly included an action involving a blue scrub brush, misidentifying it as a toy. SelfReVision successfully removed this unnecessary step. These examples highlight how SelfReVision enhances plan precision by correcting both omissions and errors, leading to more reliable task execution.

5 Conclusion

We showed that SelfReVision, a self-improvement framework for vision-language procedural planning, can significantly boost the performance of small models through iterative self-critiquing and refinement.

Limitations

While our method demonstrates promising results for low-level procedural planning in small-scale VLMs, it is not without limitations.

A primary limitation of SelfReVision is its increased inference cost. Unlike the SFT approaches that generate a complete plan in a single forward

pass, SelfReVision requires iterative refinement across multiple calls, averaging around 8 inference steps per example. This iterative process enables more accurate and grounded reasoning, but may pose challenges for latency-sensitive or real-time applications.

Second, our self-improvement strategy assumes that the model can recognize and correct its own planning errors during training. However, if the model's internal reward signal or critique mechanism is flawed, this could reinforce incorrect behaviors or lead to overfitting on superficial plan heuristics. Although we do see improvement in all models tested, a weaker model might not benefit from the same method.

Lastly, currently we only experiment with added visual inputs and do not incorporate other potentially useful modalities such as robot proprioception, or tactile feedback. This unimodal design limits the method's ability to adapt to multimodal real-world scenarios where contextual or embodied cues are critical for accurate planning. It would be interesting for future work to attempt to incorporate more versatile type of information in the self-critiquing loop.

Acknowledgement

This work was supported by funding from the Army Research Lab.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario M Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, and 24 others. 2022. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, and 16 others. 2024. Paligemma: A versatile 3b vlm for transfer. *Preprint*, arXiv:2407.07726.

Faeze Brahman, Chandra Bhagavatula, Valentina Pyatkin, Jena D. Hwang, Xiang Lorraine Li, Hirona Jacqueline Arai, Soumya Sanyal, Keisuke Sakaguchi, Xiang Ren, and Yejin Choi. 2023. Plasma: Making small language models better procedural knowledge models for (counterfactual) planning. *ArXiv*, abs/2305.19472.

Robert L. Brennan and Dale J. Prediger. 1981. Coefficient kappa: Some uses, misuses, and alternatives. *Educational and Psychological Measurement*, 41(3):687–699.

Hongyi Chen, Yunchao Yao, Ruixuan Liu, Changliu Liu, and Jeffrey Ichnowski. 2024. Automating robot failure recovery using vision-language models with optimized prompts. *Preprint*, arXiv:2409.03966.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. 2025. Navila: Legged robot vision-language-action model for navigation. *Preprint*, arXiv:2412.04453.

Kanzhi Cheng, Yantao Li, Fangzhi Xu, Jianbing Zhang, Hao Zhou, and Yang Liu. 2024. Vision-language models can self-improve reasoning via reflection. *Preprint*, arXiv:2411.00855.

Yihe Deng, Pan Lu, Fan Yin, Ziniu Hu, Sheng Shen, Quanquan Gu, James Zou, Kai-Wei Chang, and Wei Wang. 2024. Enhancing large vision language models with self-training on image comprehension. In *Advances in Neural Information Processing Systems*, volume 37, pages 131369–131397. Curran Associates, Inc.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, and 3 others. 2023. Palm-e: An embodied multimodal language model. In *arXiv preprint arXiv:2303.03378*.

Yunhao Fang, Ligeng Zhu, Yao Lu, Yan Wang, Pavlo Molchanov, Jan Kautz, Jang Hyun Cho, Marco Pavone, Song Han, and Hongxu Yin. 2024. Vila^2: Vila augmented vila. *Preprint*, arXiv:2407.17453.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alexa Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, A. Doucet, Orhan Firat, and Nando de Freitas. 2023. Reinforced self-training (rest) for language modeling. *ArXiv*, abs/2308.08998.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *EMNLP 2023 - 2023 Conference on Empirical Methods in Natural Language Processing, Proceedings*, EMNLP 2023 - 2023 Conference on Empirical Methods in Natural Language Processing, Proceedings, pages 1051–1068. Association for Computational Linguistics (ACL).

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *International Conference on Machine Learning*.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022. Inner monologue: Embodied reasoning through planning with language models. *CoRR*, abs/2207.05608.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *ArXiv*, abs/2402.02716.

Muhammet Ilaslan, Ali Koksal, Kevin Qinghong Lin, Burak Satar, Mike Zheng Shou, and Qianli Xu. 2024. Vg-tvp: Multimodal procedural planning via visually grounded text-video prompting. In *AAAI Conference on Artificial Intelligence*.

Jaehun Jung, Peter West, Liwei Jiang, Faeze Brahman, Ximing Lu, Jillian Fisher, Taylor Sorensen, and Yejin Choi. 2024. Impossible distillation for paraphrasing and summarization: How to make high-quality lemonade out of small, low-quality model. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4439–4454, Mexico City, Mexico. Association for Computational Linguistics.

Namasivayam Kalithasan, Himanshu Gaurav Singh, Vishal Bindal, Arnav Tuli, Vishwajeet Agrawal, Rahul Jain, Parag Singla, and Rohan Paul. 2022. Learning neuro-symbolic programs for language guided robot manipulation. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7973–7980.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. 2024a. *CogACT: A Foundational Vision-Language-Action Model for Synergizing Cognition and Action in Robotic Manipulation*.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. 2024b. Hamster: Hierarchical action models for open-world robot manipulation. In *CoRL 2024 Workshop on Language and Robot Learning: Language as an Interface*.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023a. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023b. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE.

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

Yujie Lu, Pan Lu, Zhiyu Chen, Wanrong Zhu, Xin Eric Wang, and William Yang Wang. 2023. Multimodal procedural planning via dual text-image prompting. *ArXiv*, abs/2305.01795.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. 2025. A survey on vision-language-action models for embodied ai. *Preprint*, arXiv:2405.14093.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang,

Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

Amir Moslemi, Anna Briskina, Zubeka Dang, and Jason Li. 2024. A survey on knowledge distillation: Recent advancements. *Machine Learning with Applications*, 18:100605.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Ting Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Presents VirtualHome and proposes a model to predict programs from videos or descriptions. The predicted program is finetuned using RL to be executable in the simulator.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. 2025. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *ArXiv*, abs/2502.19417.

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

Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, Silvio Savarese, Hyowon Gweon, Jiajun Wu, and Li Fei-Fei. 2022. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 477–490. PMLR.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, and 89 others. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Guande Wu, Huan Song, Yawei Wang, Qiaojing Yan, Yijun Tian, Lin Lee Cheong, and Panpan Xu. 2025. Sdrt: Enhance vision-language models by self-distillation with diverse reasoning traces. *Preprint*, arXiv:2503.01754.

Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Dou, Wenyu Zhan, Xiao Wang, Rui Zheng, Tao Ji, Xiaowei Shi, Yitao Zhai, Rongxiang Weng, Jingang Wang, Xunliang Cai, Tao Gui, and 5 others. 2024. Enhancing llm reasoning via critique models with test-time and training-time supervision. *CoRR*, abs/2411.16579.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2023. Self-evaluation guided beam search for reasoning. In *Advances in Neural Information Processing Systems*, volume 36, pages 41618–41650. Curran Associates, Inc.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *ArXiv*, abs/2402.13116.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. 2024a. Rlcd: Reinforcement learning from contrastive distillation for language model alignment. *Preprint*, arXiv:2307.12950.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023. Mm-react: Prompting chatgpt for multimodal reasoning and action. *ArXiv*, abs/2303.11381.

Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2024b. Guiding long-horizon task and motion planning with vision language models. *Preprint*, arXiv:2410.02193.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React:

Synergizing reasoning and acting in language models. *International Conference on Learning Representations (ICLR)*.

Xiao Yu, Baolin Peng, Michel Galley, Jianfeng Gao, and Zhou Yu. 2024. Teaching language models to self-improve through interactive demonstrations. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5127–5149, Mexico City, Mexico. Association for Computational Linguistics.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Rowan Armstrong, Eric Tzeng, and Lerrel Pinto. 2020. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning (CoRL)*. Project Website, PDF available online.

Min Zhang, Jianye Hao, Xian Fu, Peilong Han, Hao Zhang, Lei Shi, Hongyao Tang, and Yan Zheng. 2024. Mfe-etp: A comprehensive evaluation benchmark for multi-modal foundation models on embodied task planning.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *Preprint*, arXiv:2304.10592.

A Experimental Details

In this section, we provide full details of the experimentation used in this paper. We start with implementation of our method Appendix A.1, and then discuss the experimental setup of both the procedural planning Appendix A.2 and embodied agents Appendix A.3.

A.1 Method Implementation

Training Data We used a subset of images from the Places365 Dataset (Zhou et al., 2017), which contains real-world scenes categorized by location type (e.g., airport lounge, kitchen, barn). This dataset originally contains 2.5 million images which are categorized into 205 types of scenes (e.g. barn, living room, beauty salon). Some of these categories were not conducive to our experiments, specifically ones that might not allow for many tasks to be done (e.g. barndoor, batters box, ice shelf). To determine which categories to use, we had two researchers independently rate all 205 categories based on perceived eligibility to the task of procedural planning on a 4-point likert scale (1 = best category, 4 = worst category). We then included all categories which had an average score of 1.5. This resulted in the following diverse 55 categories:

- **Places365 Categories**: airplane cabin, airport terminal, apartment building outdoor, aquatic theater, arcade, archaeological excavation, archive, army base, art gallery, art studio, atrium public, banquet hall, bar, barn, basement, bathroom, bazaar indoor, beach house, biology laboratory, bookstore, chemistry lab, childs room, classroom, clothing store, coffee shop, dinette home, dorm room, florist shop indoor, florist shop outdoor, gallery, game room, gymnasium indoor, hardware store, home office, home theater, hospital, hospital room, hotel room, kindergarten classroom, kitchen, kitchenette, laundromat, living room, lobby, nursery, office, pharmacy, playroom, pub indoor, reception, recreation room, repair shop, restaurant kitchen, storage room, utility room.

We aimed to based our dataset on a diverse range of real-world images, including both indoor and outdoor scenes.

Then, within each category there is a wide range of types of images. Since this dataset uses images from a wide range of online sourced, not all the

images are of the same quality. For our task, we wanted to have scenes which were clear, easy to see, and not too focused on one object or too broad to not be able to have tangible tasks. Therefore, we choose to filter the images based on the following criteria:

- **Too Blurry**: slight blurriness is acceptable if objects remain identifiable, but excessively blurry images should be excluded.
- **Too Dark**: some darkness is acceptable as long as objects can still be discerned. However, images that are too dark to identify objects should be filtered out.
- **Too Zoomed-In/Too Zoomed-Out**: images that are overly focused on a single detail (e.g., close-ups of flowers or a single individual) and lack broader environmental context should be excluded./images taken from too far away, like more than 100 feet away, or those that primarily capture abstract landscapes, making it difficult to infer meaningful tasks specific to the environment, should be filtered out

We did this filtering automatically using GPT-4o (OpenAI et al., 2024) by prompting. The exact prompt can be see in prompt 1. In total we randomly selected 51997 (1000 images per category) images, resulting in 35619 final images after filtering.

Next, we took each of these filtered images and again prompted GPT-4o to generate a plausible user-input (see prompt 2). This resulted in a final dataset of $n = 107013$ image/user-input pairs for training.

Prompt 1. *You are evaluating an image to decide whether it should be filtered out for data generation purposes. An ideal image should provide clear environmental context for robots, as these images will be used to generate a list of tasks that robots can perform based on the given situation. Specifically, images should be filtered out if they meet any of the following criteria: 1) too blurry (slight blurriness is acceptable if objects remain identifiable, but excessively blurry images should be excluded.), 2) too dark (some darkness is acceptable as long as objects can still be discerned. However, images that are too dark to identify objects should be filtered out.), 3) too zoomed-in (images that are overly focused on a single detail (e.g., close-ups of flowers or a single individual) and lack broader environmental context should be excluded.), 4) too far-out (images taken from too far away, like more than 100 feet away, or those that primarily capture abstract landscapes, making it difficult to infer meaningful tasks specific for the environment, should be filtered out).*

Please provide feedback for each criterion and the overall decision in JSON format as shown in the example below: ‘"blurry":"blurry/ok","darkness":"too dark/quite dark/slightly dark/ok", "zoomed-in": "too zoomed-in/somewhat zoomed-in/ok", "far-out":"too far-out/somewhat far-out/ok", "decision":"keep/filter"‘

Prompt 2. *Given an image generate 3 plausible user inputs from someone in the image directed at a robot, which would then cause the robot to do a task. The user inputs can be statements or questions.*

Also, for each input, generate a list of high-level steps for the robot to finish the task. Make sure the high-level steps are specific to the setting in the image.

Lastly, for each input, generate a short response by the robot that indicates what it plans to do.

Do not mention the image or picture. The user inputs should be very different from each other and specific to the scene. Separate the high-level steps using "|". Respond strictly in JSON format with 9 keys: ‘User_Input1’, ‘Steps1’, ‘Robot_Response1’,. . . , ‘User_Input3’, ‘Steps3’, ‘Robot_Response3’. Do not use any markdown formatting or code block symbols (such as triple backticks).

*** Multiple like-version of this prompt was used, see Github code for full list***

Self-Distillation/Improvement In order to generate the high-level plans (the labels of our training data) we used the base model itself through prompting in a process called self-distillation. First, we use a general prompt to get an initial plan p_0 (see prompt 3.

However, given the weak nature of the base model, this prompt is not going to be well grounded to the given scene. Therefore, we use a series of self-critique, self-revise, and self-evalute prompts to generate a better final plan. First, we self-critique the initial plan using an open-ended prompt $\text{Crit}(p_0)$, see prompt 4. Then, we used the output from this prompting along with the original plan p_0 to revise the original plan $\text{Rev}(p_0, \text{Crit}(p_0)) = p_1$,

see prompt 5. Lastly, we prompted the base model to verify if the revised plan is better than the original plan using prompt 6 $\text{Ver}(p_0, p_1)$.

Prompt 3. *You are writing instructions for a robot in the image. Make a detailed plan which responds to the users input. You can only use the items you see in the given image and must make your plan specific to this setting.*

You should respond with only the numbered plan which starts with "<plan>" and ends with "</plan>". No other text should be outputted. Do not use any markdown formatting, code block symbols (such as triple backticks), headings, summaries, or nested bullet points

User Input: "{user_input}"

Prompt 4. *You are reviewing a high-level plan for a robot based on a user request and an image of the environment.*

Your goal is to identify critical flaws, gaps, or missed opportunities that would significantly improve the plan's feasibility, clarity, or alignment with the depicted environment. Focus on major missing steps, unrealistic assumptions, or vague actions that reduce the quality of the plan. Avoid nitpicking or commenting on minor stylistic issues.

Ground your feedback in the visual context and user intent. Prioritize issues that would materially impact the robot's ability to execute the task successfully.

Output a clean, single-level numbered list of feedback enclosed between <critic> and </critic>. Each item should describe one clear issue or suggestion for meaningful improvement.

Do not suggest rewordings or edits—focus only on diagnosing problems.

User Input: "{user_input}"

Current Plan: "{current_plan}"

Prompt 5. *You are revising a high-level robot plan based on critical feedback, the user's request, and an image of the environment.*

Use the feedback to identify key flaws and address them with substantive improvements. Focus on clarity, feasibility, and grounding the plan in the actual visual context. Prioritize corrections that enable the robot to effectively and realistically complete the task.

*Make **meaningful changes**, not surface-level edits. Omit redundant or overly detailed instructions that don't improve execution. Avoid speculative details unless they're clearly justified by the visual context.*

Output a clean, single-level numbered list of steps enclosed between <plan> and </plan>. Do not include titles, nested lists, extra commentary, or any formatting besides the numbering.

User Input: "{user_input}"

Current Plan: "{current_plan}"

Feedback: "{criticism}"

Prompt 6. *You are evaluating two sets of instructions for a robot in the image. You will be given a user input and two high-level plans. Compare the two plans and respond with "yes" if Plan 2 better fulfills the user request than Plan 1; otherwise, respond with "no". Good plans generally use only items visible in the image and are specific to the setting shown. A better plan more effectively uses only items visible in the image and is more specific to the setting shown. It also demonstrates stronger coverage, more logical order, greater completeness, and better grounding in the image. Do not use any markdown formatting or code block symbols (such as triple backticks)."*

User Input: "{user_input}"

Plan 1: "{initial_plan}"

Plan 2: "{revised_plan}"

Training We used a diverse range of base models to experiment with SelfReVision Qwen-2.5-VL-Instruct (3B, 7B, 32B, 72B) (Bai et al., 2025) and Gemma 3 (4B, 12B, 27B) (Team et al., 2024). We performed supervised fine-tuning of the base models using plans generated with SelfReVision During training, all models were cast to the torch.bfloat16 data type and trained for 4 epochs. The best model was selected based on cross-entropy loss on a development set consisting of 100 randomly held-out examples from the training data. Final evaluation results (win rates) were computed on a separate set of 100 held-out samples. We experimented with three learning rates (1e-5, 3e-5, and 5e-5) for each model and report results for the best-performing one. Weight decay was fixed at 0.01, and the maximum number of tokens was set to 500 for all models.

A.2 Goal-Based Procedural Planning Details

In this section we outline the experimental details for the goal-based procedural planning experiments.

Evaluation Dataset We evaluated our method on both real-world setting and simulation setting datasets. For the real-world setting, we used a

randomly selected held-out test set of $n = 100$ image and user-input pairs from our training data. These images were sampled from the Places365 Dataset (Zhou et al., 2017), and the corresponding user inputs were generated using GPT-4o (OpenAI et al., 2024). See Appendix A.1 for full details.

For the simulation setting, we used a modified version of the MFE-ETP benchmark dataset (Zhang et al., 2024), which consists of $n = 100$ image and user-prompt pairs drawn from the popular procedural simulation environments VirtualHome (Puig et al., 2018) and BEHAVIOR-100 (Srivastava et al., 2022). This dataset was created as a challenging benchmark for embodied reasoning and procedural planning. However, for some of the original MFE-ETP samples, there are multiple images of the initial conditions which might be needed to create a plan for the given task. Since, we want to focus on only one image for a user-input, we hand-selected teh best image for the given task. If no image captured enough information to complete the task, we randomly selected an image and wrote a new task. The full list of the $n = 100$ choosen images and tasks can be found on our github.

Baselines To demonstrate the effectiveness of SelfReVision, we first compare the refined plans to the initial plans generated by the models using few-shot prompting. We also evaluate responses from other baselines such as GPT-4o (representing a powerful large model) (OpenAI et al., 2024), PaliGemma (a domain-specific model trained for planning) (Beyer et al., 2024), a Basic Distillation (inspired by Shi et al. (2025), we use a more detailed self-distillation prompting that includes instructions for physical and spatial grounding) and best-of-N (an inference-time algorithm that generates multiple outputs and selects the best one). The prompts and examples provided to GPT-4o and PaliGemma match those given to the base. The prompt for the Basic Distillation technique can be found in prompt 8.

For the best-of-N baseline, we use $N = 5$: we sample five different plans with a temperature of 0.5, followed by a final inference step to select the best plan among them. This setup approximately matches the number of additional inferences made by both SelfReVision and the baseline.

prompt 7 shows the exact prompt used to do few-shot generation with baselines.

Prompt 7. *You are writing instructions for a robot in the image. Make a detailed plan which responds to the users input. You can only use the items you see in the given image and must make your plan specific to this setting. You should respond with only the numbered plan and no other text should be outputted. Do not use any markdown formatting or code block symbols (such as triple backticks).*

Example 1 User Input: Hmm, I don't think the time on that clock is correct. Plan: 1. Navigate to the Clock 2. Grab the Clock 3. Adjust the Time to 12:15 4. Return the Clock

Example 2 User Input: Can you make my drink colder? Plan: 1. Navigate to the Fridge 2. Open the Freezer Door 3. Locate the Ice Tray 4. Collect the Ice 5. Close the Freezer Door 6. Navigate back to the Person 7. Put the Ice in the Drink

Example 3 User Input: Can you hang this picture for me? Plan: 1. Pick up the Hammer and Nail 2. Insert Nail into the Wall with Hammer 3. Put Down the Tools 4. Pick up Picture 5. Hang the Picture

User Input: user_input

Plan:

Prompt 8. *You are writing a detailed plan for a robot to carry out a user request based on a given image of the environment.*

Your plan must: - Use only the objects and elements visible in the image. - Be grounded in the physical layout shown—consider spatial relationships, object accessibility, reachability, and possible obstructions. - Avoid assumptions that cannot be confirmed visually. - Break the task into clear, atomic steps that the robot can execute. - Consider practical challenges the robot might face (e.g., needing to navigate around an obstacle or pick up an object from a specific angle).

*Do **not** output any explanations, justifications, or summaries. Your response should contain only the numbered list of steps, enclosed between <plan> and </plan>.*

Begin your response with "<plan>" and end it with "</plan>". Do not include markdown formatting, headings, code block symbols, or any extra text.

User Input: user_input

Plan:

Ablation Study Details To evaluate the contribution of each component in SelfReVision self-refinement loop, we conducted a series of ablation experiments by selectively removing individual stages. Table 3 presents the ablation results on both the PLACES and SIMULATION datasets,

averaged across the seven VLMs. We compare four configurations: the full CRV (Criticize-Revise-Verify) pipeline, CR (Criticize-Revise), RV (Revise-Verify), and R (Revise-only).

prompt 9 shows the revision prompt for variants that do not go through the self-criticism process (RV and R).

Prompt 9. *You are revising a high-level plan for a robot. You will be given a user's input and the current plan. Your task is to revise and improve the plan.*

When revising: 1. Make sure to use only objects visible in the image 2. Provide a step-by-step plan specific to the setting 3. Address all aspects of the user input 4. Ensure logical ordering of actions 5. Add spatial details where needed 6. Ensure all actions are feasible in the environment shown

Respond only with the revised, numbered steps which starts with "<plan>" and ends with "</plan>". Do not include any additional text. Do not use markdown formatting or code block symbols (such as triple backticks).

User Input: user_input

Current Plan: current_plan

Evaluation Methodology and Other Details In line with prior work (Brahman et al., 2023; Huang et al.), we evaluate procedural plans using the following five criteria:

- **Coverage** — How well the plan addresses the user's input.
- **Ordering** — Whether the plan follows a coherent and logical sequence.
- **Completeness** — Whether the plan is sufficiently detailed and informative.
- **Image Groundedness** — Whether the plan is plausible given the specific visual scene.
- **Overall Quality** — The overall effectiveness and appropriateness of the plan.

We include the *Image Groundedness* criterion to reflect the visual nature of our model: unlike prior work focused solely on language models (LLMs), our goal is to develop a vision-language model (VLM) that generates plans tailored to specific images.

Given the strong performance of LLMs-as-judges (Zheng et al., 2023), we use GPT-4o (OpenAI et al., 2024) as an automated evaluator via prompting. See Appendix B for full details on validation of this method. The prompt we used to evaluate can be seen in prompt 10.

Prompt 10. *You will be given an image of a setting, a user input and a corresponding plan with high-level steps that can be used by a robot to respond to the user input in that setting. Only output a valid json (python dictionary) and keep any explanation brief < 10 words. Your task is to evaluate the plan based on the following five criteria:*

*Coverage (Does the plan fully address the user input?)***
*- **5 (Definitely): The plan thoroughly addresses all aspects of the user input without omissions.*
*- **4 (Mostly): The plan covers the main points of the user input, but might miss a few minor details.*
*- **3 (Somewhat): The plan addresses some aspects of the user input, but not comprehensively.*
*- **2 (Slightly): The plan barely touches on the user's input and misses several key points.*
*- **1 (Not at all): The plan fails to address the user input or is irrelevant.*

*Ordering (Is the plan well-ordered?)***
*- **5 (Definitely):** The ordering does not need any changes.*
*- **4 (Mostly):** The ordering is generally good, but there might be a few minor adjustments.*
*- **3 (Somewhat):** I could see reordering some of these, but it would be more of a stylistic change.*
*- **2 (Slightly):** The ordering could use some improvements, but it's not entirely bad.*
*- **1 (Not at all):** Ordering is bad or nonsensical.*

*Completeness (Is the plan complete and informative?)***
*- **5 (Definitely):** The plan provides a complete and informative picture of what needs to be done to respond to the user input.*
*- **4 (Mostly):** The plan is mostly complete and informative, with only a few minor gaps.*
*- **3 (Somewhat):** The steps are somewhat general, but overall you get what you need. You might need a few minor details.*
*- **2 (Slightly):** The plan is missing several key details and is not fully clear.*
*- **1 (Not at all):** The plan is really bland and dominated by unnecessary, irrelevant, and/or repetitive steps, or key steps are missing.*

*Image Grounded (Can this plan be carried out in the specific setting shown in the image?)**
– **5 (Definitely): All objects and actions mentioned are clearly present in the image; the plan is specific to the setting seen in the image.
– **4 (Mostly): The plan makes sense for the setting seen in the image, with only minor mismatches (e.g., one object might be assumed but not shown, or include vague actions to be done in the image presented).
– **3 (Somewhat): The plan is partially grounded in the setting shown in the image, but some steps rely on questionable assumptions about what's available or possible to be done.
– **2 (Slightly): Several actions or objects don't appear to match the specific setting in the image, making the plan hard to execute as described.
– **1 (Not at all): The plan feels unrealistic or unrelated to the specific setting in the image—objects are used that are not in the image, actions are implausible or vauge, or it seems like the setting was ignored entirely.

*Overall (Is the plan overall good?)**
- **5 (Definitely):** The plan is overall good. A good plan should be well-ordered, complete, and contain no repetitive or unnecessary steps.
- **4 (Mostly):** The plan is mostly good. It's generally well-organized and complete but could use some improvements in detail or clarity.
- **3 (Somewhat):** The steps are somewhat general, but overall you get what you need.
- **2 (Slightly):** The plan is lacking in key details, and some steps feel unnecessary or unclear, but it somewhat meets the requirements.
- **1 (Not at all):** The plan is really bland and not good with repetitive or unnecessary steps.
—
Example 1 Input
"user_input": "Can you take my picture with this background?",
"high_level_plan": [
"1. Navigate to the Arch",
"2. Position at the Ideal Angle",
"3. Adjust Camera Settings",
"4. Capture the Panoramic Photo"
]
Example 1 Output
"Coverage": 5,
"Coverage Explanation": "The plan is completely relevant to the user input.",
"Ordering": 5,
"Ordering Explanation": "The steps are in the correct order.",
"Completeness": 4,
"Completeness Explanation": "The plan is mostly complete but lacks specific details about how to adjust the settings.",
"Image Grounded": 4,
"Image Grounded Explanation": "The plan includes only objects in the setting, but it maybe be hard to navigate through the rocks without more directions.",
"Overall": 4,
"Overall Explanation": "The plan is mostly good with minor gaps in detail."

Example 2 Input
"user_input": "I'm going on a roadtrip, can you grab me a snack?",
"high_level_plan": [
"1. Navigate to the Fridge",
"2. Open the Fridge",
"3. Grab a Grape",
]
Example 2 Output
"Coverage": 4,
"Coverage Explanation": "Although the plan does get food, one grape might not be enough food for a roadtrip.", "Ordering": 5,
"Ordering Explanation": "The steps are in the correct order.",
"Completeness": 2,
"Completeness Explanation": "The plan does not bring the food to the human.",
"Image Grounded": 5,
"Image Grounded Explanation": "The plan includes objects in the setting.",
"Overall": 3,
"Overall Explanation": "The plan is only slightly address the user input but does not complete it."
Respond strictly in JSON format with the key "Coverage", "Coverage Explanation", "Ordering", "Ordering Explanation", "Completeness", "Completeness Explanation", "Image Grounded", "Image Grounded Explanation", "Overall", and "Overall Explanation". Do not use any markdown formatting or code block symbols (such as triple backticks).

A.3 Embodied Agents Details

In our second set of experiments we aimed to see how our SelfReVision ight result in better down-

stream performance for embodied agents. We used two simulated experiments to test this hypothesis.

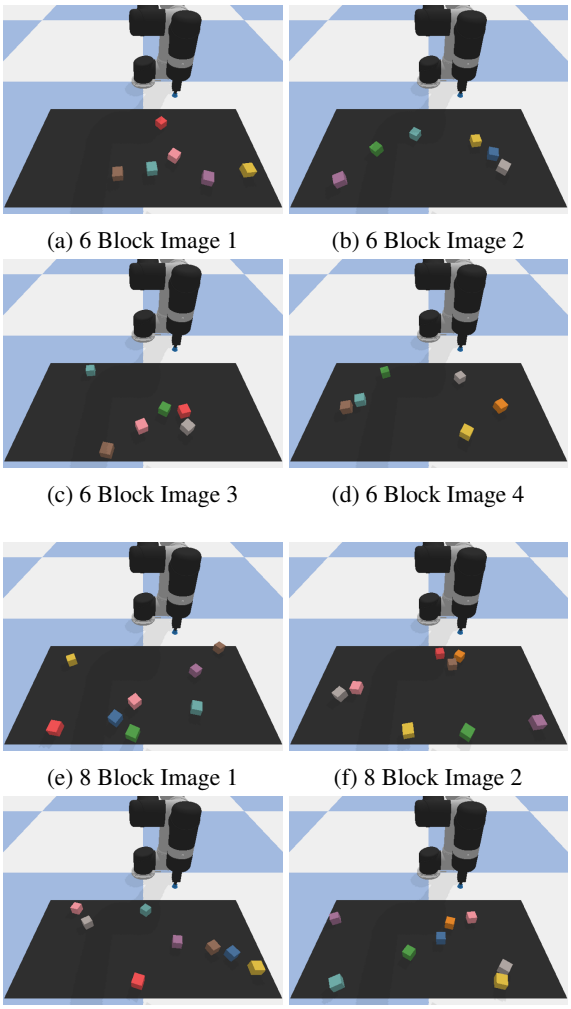
Evaluation Set We used two distinct simulation environments for evaluation: (1) block manipulation tasks from Ravens (Zeng et al., 2020), and (2) complex, hierarchical tasks from HAMSTER (Li et al., 2024b). For the Ravens environment, we curated 14 unique manipulation goals, each paired with 8 different initial block configurations involving 6 or 8 blocks—yielding a total of $n = 112$ samples. Each configuration had blocks of unique colors. Figure 6 shows the 8 individual block scenes and here is the full list of 14 goals are:

- Form a shape of an uppercase X with the blocks.
- Form a shape of an uppercase O with the blocks.
- Form a shape of an uppercase Y with the blocks.
- Form a shape of an uppercase V with the blocks.
- Form a shape of an uppercase W with the blocks.
- Form a diagonal line.
- Form two diagonal lines.
- Form two vertical lines.
- Form two horizontal lines.
- Create a smiley face.
- Create a frowning face.
- Form a shape of triangle with the blocks.
- Form the shape of a house.
- Form a rainbow.

.

For the hierarchical setting, we designed 10 realistic task scenarios across three environments—kitchen, workshop, and office—each involving a high-level task (e.g., "Pack items for a children's lunch"). Appendix A.3 shows the 10 realistic task with corresponding goals.

Metric For the simulated pick-and-place environment, we run each plan using a code-as-policies simulator (Liang et al., 2023b) which generated a static image for each step. Then, a human rater evaluated the final configuration, judging whether the plan achieved the stated goal. We then calculated the average number of samples where the code-as-policy successfully ran the plan and achieved the final state. For the real-world settings, we used Li et al. (2024b) to generate a trace path for each step in each generated plan. Then, a human raters assessed whether each individual step was completed



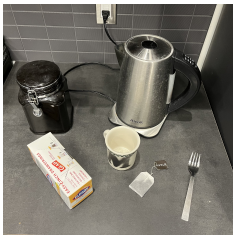
(a) 6 Block Image 1 (b) 6 Block Image 2

(c) 6 Block Image 3 (d) 6 Block Image 4

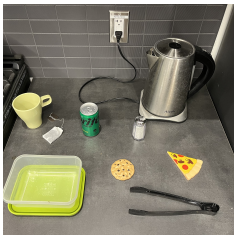
(e) 8 Block Image 1 (f) 8 Block Image 2

(g) 8 Block Image 3 (h) 8 Block Image 4

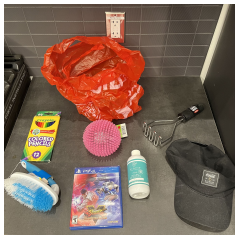
Figure 6: Eight initial scenes used for the block manipulation task.



(a) There is water in the kettle. Can you make me a cup of tea?

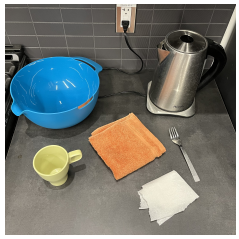


(b) I need to pack a lunch for my kid in the clear tupperware. Only include foods a kid would like.

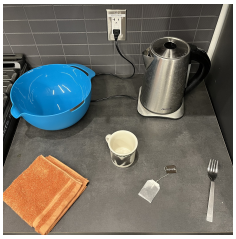


(c) Pack items a kid would like into the red bag. If you do not know what an object is, don't include it.

(d) Please organize the trash into the right bins. Note that the paper towel is not used.



(e) Can you place objects in the bowl in order to clean the counter?



(f) Can you clean the counter to look less messy? No need to wipe the counter, just consolidate all the things.

(g) Can you declutter the workshop table?

(h) Hammer the nail in the wood.

(i) What is the easiest way to water the plant?

(j) Can you tell me how to charge my phone with what is in this setting?

Figure 7: Images used in the real-world simulation experiments with corresponding goals.

successfully by the generated trace. We then indicated the success rate, which is the number of traces that were deemed successful for a step divided by all steps. Note, we did not include steps that would not result in a trace such as "Move to <object>".

Baselines We compared the plans generated by SelfReVision with the initial base plan created by the model. For this task we used only Gemma 12B and 27B (Team et al., 2024).

A.4 Software

We used Python 3.12.9, Pytorch 2.6.0, and HuggingFace Transformers 4.51.0.All code is licensed under the Apache License 2.0.

A.5 Hardware

All experiments were run on a cluster with 24 NIVIDIA A100 GPUs with 80B memory. For most inference jobs we used one GPU but for 72B models we needed two GPUs. For supervised fine-tuning, we used on GPU for Qwen 3B, two GPUs for Gemma 4B and Qwen 7B, four GPUs for Gemma 12B. The training for four epochs took about two days.

A.6 Artifact Terms of Use

Places365 (Zhou et al., 2017): MIT License

B LLM-as-Judge Analysis

In our study, we used LLM-as-Judge as the main metric for comparison between plans. In this section, we outline our process for evaluating the robustness of using an LLM instead of human raters.

We did a test on a sample of $n = 60$ examples where we evaluated the quality of two robot plans: Plan 0 and Plan n using both human annotators and GPT-4o as an LLM-as-a-Judge. To reduce positional bias during annotation, we randomly assigned these two plans to anonymized labels Plan A and Plan B for each sample shown to human raters. Each plan pair (Plan A and Plan B) is scored on five criteria: Coverage, Ordering, Completeness, Image Groundedness, and Overall. The full annotation instruction can be found in Figure 8.

To measure agreement, we collected annotations from three human annotators and three GPT-4o runs at temperature 0.6. Model outputs were generated using identical prompts and image inputs, with variation arising only from randomized sampling. This setup allowed us to capture inter-model variability due to sampling while maintaining a

consistent evaluation protocol.

We chose the Brennan-Prediger coefficient as our agreement metric because it adjusts for chance agreement and handles categorical labels (Plan A", Plan B", or "Tie"). Unlike raw accuracy, it remains robust under label imbalance and is well-suited for comparing multiple raters with potentially different labeling tendencies.

We report Brennan-Prediger agreement coefficients (Brennan and Prediger, 1981) between all pairs of raters. The top-level results are summarized below, where we report the mean pairwise agreement between:

1. Human-Human pairs (3 combinations)
2. Model-Model pairs (3 combinations)
3. Human-Model pairs (9 combinations)

These are computed for each of the five evaluation criteria, and the table below reflects averages across the respective pairings.

To better understand how often annotators reached full consensus, we measured the percentage of plan pairs where all three human annotators selected the same label:

- **Coverage:** 60% agreement
- **Ordering:** 43% agreement
- **Completeness:** 40% agreement
- **Image Groundedness:** 50% agreement
- **Overall:** 27% agreement

Model-model agreement reflects intra-model consistency under sampling variation. The high model-model agreement across criteria (e.g., 0.94 for Ordering and Overall, 0.89 for Coverage) indicates that GPT-4o produces stable and repeatable judgments across independent runs. Moreover, model-human agreement scores are consistently competitive with human-human agreement—e.g., 0.55 vs. 0.58 for Image Groundedness, 0.56 vs. 0.43 for Completeness, and 0.60 vs. 0.74 for Coverage. These results suggest that GPT-4o is not only internally consistent but also meaningfully aligned with human judgment, supporting its use as a reliable automated judge in comparative plan evaluation tasks.

C Information About Use Of AI Assistants

In this project, AI assistants were used for results visualization code (e.g., figures and tables) and for minor writing edits.

You will be given a user input and two corresponding plans (Plan A and Plan B) with high-level steps that can be used by a robot to respond to the user input in a specific setting. I will also provide an image of the setting when available.
Your task is to evaluate which plan is better based on the following criteria:
Coverage (Does the plan fully address the user input?) - Does the plan thoroughly address all aspects of the user input without omissions? - Does the plan cover the main points of the user input, or does it miss details?
Ordering (Is the plan well-ordered?) - Is the sequence of steps logical and efficient? - Would any reordering of steps improve the plan?
Completeness (Is the plan complete and informative?) - Does the plan provide a complete picture of what needs to be done? - Are the steps specific and detailed enough? - Are there any gaps in the plan?
Image Groundedness (Can this plan be carried out in the specific setting shown in the image?)** - Are all objects and actions mentioned clearly present or possible in the given setting in the image? - Is the plan specific and well grounded to the setting seen in the image?
Overall Assessment - Considering all criteria above, which plan is better overall?

Figure 8: The instruction given to the human annotators

Criterion	Human-Human	Model-Model	Human-Model
Coverage	0.74	0.89	0.61
Ordering	0.48	0.94	0.46
Completeness	0.43	0.84	0.56
Image Grounded	0.58	0.81	0.55
Overall	0.26	0.94	0.44

Table 5: Brennan-Prediger agreement coefficients for human-human, model-model, and human-model rater pairs, averaged across all combinations and 60 plan comparison samples. GPT-4o was run with temperature 0.6.