APIVOT: Adaptive Planning with Interleaved Vision-Language Thoughts

Emily Jin **Joy Hsu** **Yiqing Xu** **Weiyu Liu**[†] **Nick Haber**[†] **Jiajun Wu**[†]
Stanford University

Abstract

Long-horizon robot planning requires joint reasoning over semantic task structure and geometric feasibility. While language supports high-level goal decomposition and action sequencing, resolving geometric constraints such as limited free space and collision avoidance requires more precise visual imagination. Crucially, reasoning demands vary within a task, necessitating models that dynamically allocate visual reasoning when needed. To this end, we propose APIVOT, a VLM-based planner that adaptively interleaves language with visual thoughts for more effective and efficient long-horizon planning. We introduce a post-training framework that teaches APIVOT to leverage language for semantics, while selectively using visual thoughts as imagined future states for internal verification of geometric feasibility. On long-horizon kitchen tasks, APIVOT outperforms general-purpose VLMs and prior planning frameworks, achieving the largest gains in spatially constrained settings. We find that APIVOT learns meaningful modality selection behavior, improving reasoning efficiency and performance under limited token budgets. [*]

1 Introduction

Long-horizon robot planning requires flexibly interleaving semantic and geometric reasoning. Consider the task: "store the leftovers in the fridge." To successfully achieve this goal, a robot must reason semantically to identify which items need to be stored, select appropriate containers, and determine a sequence of actions that satisfies prerequisites (e.g., the fridge must be open before placing anything inside). At the same time, successful execution depends on geometric constraints, such as whether the leftovers fit inside the selected containers, how the containers should be arranged inside the fridge, and whether any existing objects must be moved aside to create enough free space. These two modes of reasoning are deeply intertwined. A symbolically valid plan may still fail if the containers collide, while geometric decisions made early on can affect which actions remain feasible downstream.

Existing LLM- and VLM-based planners can reliably produce semantically plausible action sequences, but often struggle when success depends on geometric feasibility [1–8]. Prior work addresses this by coupling the model with external motion planners, feasibility checkers, or learned dynamics models [9–13]. However, these systems typically incorporate geometric feedback for replanning, which does not shape the planner's internal reasoning. Instead, we argue that an effective planner should interleave semantic and geometric reasoning itself, using language and vision as complementary modalities. Language is effective for semantic reasoning, such as task decomposition and action selection [14], but it cannot express the geometric structure needed to plan over resulting scenes in a compact, precise way. By contrast, visual representations can encode spatial layout, object shapes, and remaining free space, making them well-suited for geometric reasoning [15, 16]. Crucially, effective and efficient planning requires learning *when* to use visual thoughts based on the geometry of the scene. Placing an object in an empty refrigerator may be straightforward, while placing it in a crowded one may

[†]Equal advising.
[*]Project page: `https://emilyzjin.github.io/projects/apivot.html`

Preprint.

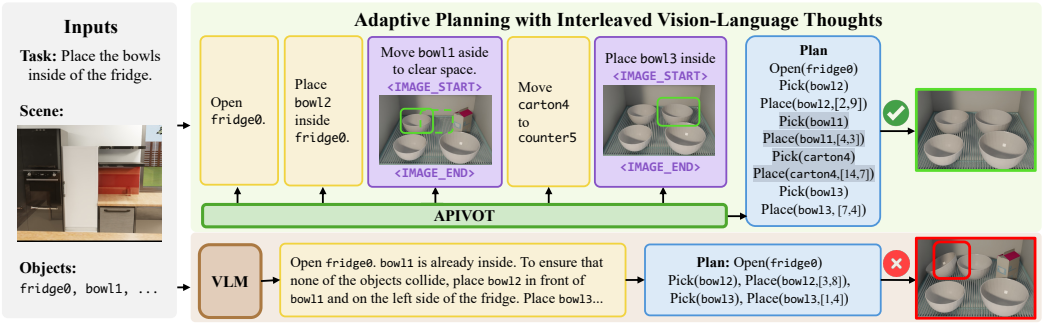


Figure 1: APIVOT (above) plans for long-horizon tasks by interleaving language reasoning with visual thoughts. While a standard VLM planner (below) reasonsing in text may produce a semantically plausible but geometrically infeasible plan, APIVOT imagines future states inside its reasoning trace. These visual thoughts reveal potential collisions or constraints before execution, enabling the planner to verify geometric constraints during planning.

require imagining the resulting layout to avoid collisions. While recent vision-language-action (VLA) models begin to incorporate visual representations for intermediate reasoning, a challenge remains in learning when to leverage vision [14–19].

To this end, we propose APIVOT, a VLM-based planner that adaptively interleaves language with visual thoughts to perform long-horizon robot planning (Figure 1). Given a task instruction and image observation of the scene, our model first produces a reasoning trace containing language and visual thoughts, before outputting a sequence of actions for execution. It learns to use language for semantic reasoning about subgoal decomposition and action ordering, and visual thoughts that represent imagined future states for verifying geometric feasibility and guiding downstream planning. Crucially, our model learns *adaptive* modality selection, choosing the appropriate modality at each planning step to jointly perform semantic and geometric reasoning.

APIVOT is trained with supervised fine-tuning (SFT) on reference traces that demonstrate successful planning and efficient modality selection behavior. These traces supervise the model to reason with visual thoughts, while the corresponding latent visual states are aligned with ground-truth image features of the scene at intermediate states. We train APIVOT under a three-stage curriculum that progressively teaches it to plan adaptively. In the first stage, the model learns to reason over provided visual thoughts. Next, it learns to autoregressively generate these visual thoughts internally and optimize them for downstream planning. In the third stage, it learns adaptive modality selection, generating visual thoughts only when they are useful for the current planning step.

We evaluate APIVOT on a suite of long-horizon planning tasks in KitchenWorlds [10]. Across all task families, APIVOT significantly outperforms existing approaches, including general-purpose VLMs, VLM-based planning frameworks, and symbolic planners. Our model achieves an average task success rate of 0.419, demonstrating an 8.1 point improvement over the top-performing baseline. In particular, APIVOT demonstrates the largest advantage in geometrically constrained settings, widening the gap from 7 to 17 points as complexity increases. Beyond improving task success, we find that our method achieves these gains with strong computational efficiency. APIVOT consistently outperforms language-only VLMs across various reasoning budgets. This suggests that visual thoughts provide a compact, expressive representation for planning, capturing spatial information that would otherwise be verbose or difficult to specify in language. We further analyze the modality selection behavior learned by APIVOT. While using visual thoughts at every step provides an upper bound on performance, APIVOT retains 91% of this performance while reducing token usage by 39%. This efficiency gain is consistent with our finding that APIVOT demonstrates a learned geometric proxy, generating visual thoughts selectively on steps requiring geometric precision.

In summary, our contributions are the following:

- We introduce APIVOT, a VLM-based planner that adaptively interleaves language for semantic reasoning and visual thoughts as imagined future states for internal geometric reasoning.
- We propose a post-training framework that progressively teaches the model to reason over, generate, and adaptively use visual thoughts during long-horizon planning.
- We demonstrate that our method improves both task success and efficiency across a diverse set of ecological, long-horizon planning tasks. APIVOT outperforms general-purpose VLMs and planning frameworks, while reducing reasoning cost through adaptive modality selection.

2 Related Work

Reasoning for long-horizon robotic manipulation. Long-horizon robotic manipulation requires reasoning over both symbolic task structure and geometric feasibility. Existing approaches can be broadly grouped into three families. Classical task and motion planning methods combine symbolic search with geometric feasibility checking or sampling [20–23]. While principled, they require hand-specified symbolic abstractions and domain models. By contrast, LLM- and VLM-based planners leverage pre-trained models to decompose natural-language instructions into action sequences, programs, or high-level plans [1–8]. These approaches offer greater flexibility, but often reason primarily at the semantic level, producing plans that are plausible in language but infeasible geometrically. A third line of work improves the geometric grounding of LLM- and VLM-based planners by integrating them with external planners, verifiers, or learned dynamics models [9–13, 24, 25]. The external modules provide geometric feedback through feasibility checks and replanning, improving execution but leaving geometry outside of the model's own reasoning process. Our work belongs to the broader class of VLM-based planners, but integrates geometric reasoning directly into planning. APIVOT generates visual thoughts within its own reasoning trace, producing imagined future states to condition subsequent decisions before the plan is produced. In doing so, geometric reasoning is an internal part of planning rather than a post-hoc verification signal.

Multimodal reasoning and vision-language-action models. While prior planning methods often introduce geometry through external verifiers or simulators, an alternative is for the planner to perform multimodal reasoning itself. Recent unified multimodal models learn shared frameworks for visual understanding and generation [26–32]. These models support general-purpose visual generation, but do not address how generation should guide long-horizon decisions. A related line of work studies multimodal chain-of-thought or latent visual reasoning, where models reason with intermediate visual representations instead of purely in text [33–42]. These approaches improve visual and spatial reasoning, but largely operate on the current observation through attention, visual marking, perception refinement, or static problem solving [35–38]. However, robot planning requires reasoning beyond the current image to planning how the scene will evolve over multiple actions. A third line of work brings multimodal reasoning into robotics through vision-language-action models and embodied foundation models [14–19]. These methods use visual intermediates, spatial representations, or multimodal context to improve action generation, but primarily target policy execution rather than high-level semantic planning. In contrast, APIVOT uses visual thoughts within a high-level planning trace, learning to use vision and language adaptively.

Adaptive and efficient multimodal reasoning. A separate line of work aims to improve models' reasoning efficiency by asking *how much* computation a model should allocate. Some methods allocate different thinking budgets, compress reasoning into latent tokens, or decide when explicit chain-of-thought is needed [41, 43]. In multimodal settings, related methods reduce the cost of visual reasoning by making the visual representation more compact, using latent visual tokens, specialist visual modules, or explicit primitives such as points and bounding boxes [36, 37, 39, 40, 42]. In embodied settings, prior work improves reasoning efficiency through lightweight training strategies or by interleaving intermediate reasoning with action prediction [14, 44]. In general, these approaches improve efficiency by making reasoning cheaper or more selectively allocated.

In robot planning, however, efficiency can also depend on *which representation* is appropriate for a given decision. Language is efficient and expressive for semantic decisions such as goal decomposition, object selection, and preconditions. Visual thoughts are more appropriate for geometric reasoning about fitting objects into constrained spaces or evaluating remaining free space, but are also more expensive [45–47]. APIVOT therefore learns adaptive modality selection. It chooses between language and visual thoughts according to the structure of the planning decision. This makes adaptivity task-grounded, improving performance while reducing unnecessary visual reasoning.

3 Vision-Language Adaptive Planning

In this section, we present APIVOT, a VLM-based planner that learns to plan by interleaving language with visual thoughts within its reasoning trace, as illustrated in Figure 2. We begin by giving an overview of the problem formulation and describing the overall planning pipeline (Section 3.1). After, we introduce our supervised fine-tuning setup (Section 3.2). Finally, we present the training

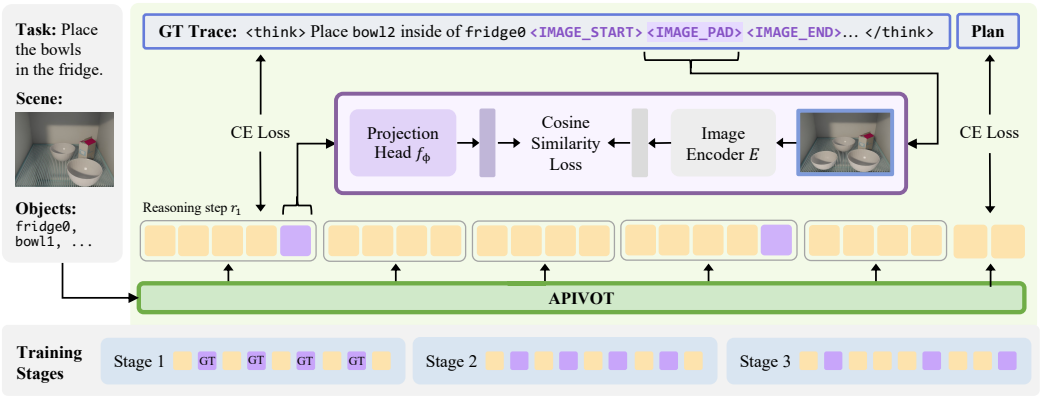


Figure 2: APIVOT produces a reasoning trace of interleaved language tokens (yellow) and visual-thoughts (purple) followed by a plan, and is trained with a three-stage SFT curriculum. In all stages, we apply teacher-forced SFT to a reference reasoning trace and plan (blue border). Stage 1 teaches the model to plan with ground-truth visual thoughts, provided at every reasoning step; Stage 2 trains it to generate visual thoughts itself by aligning them to encoded ground-truth future observations via a cosine similarity loss; and Stage 3 omits visual thoughts for some reference steps, teaching the model to plan adaptively with visual thoughts.

curriculum that teaches the model to adaptively generate visual thoughts alongside language and leverage them for planning (Section 3.3).

Problem formulation. We consider goal-conditioned planning in a Semi-Markov Decision Process (semi-MDP) with parameterized skills [48, 20]. The environment is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is a finite library of parameterized skills, and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function induced by skill execution. Each skill $a \in \mathcal{A}$ takes discrete object arguments together with optional continuous parameters. In this work, we instantiate $\mathcal{A}$ with three skills: `open(obj)`, `pick(obj)`, and `place(obj, target,` u`,` v`)`, where (u, v) is an image-space placement coordinate that is lifted to a world-frame pose using the observed depth image and the known camera extrinsics. An episode is specified by an initial state s_0 and a natural-language goal g, and terminates when g is achieved or a timeout is reached. We assume access to a dataset $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$ of successful demonstrations, where each trajectory $\tau_i = ((s_i^t, a_i^t))_t$ is annotated with the skill a_i^t applied at each step.

At each timestep t, the planner observes an image observation I_t of the current state s_t together with a list of detected objects $\mathcal{O}_t = \{(o_i, [x_i^t, y_i^t])\}$, where o_i is an object instance and $[x_i^t, y_i^t]$ is its image-space location. Detected object positions are provided as input to abstract away possible errors in object recognition and 2D localization, while preserving the need for geometric reasoning and decision-making. Concretely, the model needs to infer all such task-relevant properties from the visual observation itself, in order to reason about symbolic states, spatial relations, and feasible placements. We execute plans in a closed-loop, receding-horizon setup [49]: given $(I_t, \mathcal{O}_t)$ and the goal g, the planner outputs a remaining plan $(a_t^1, \ldots, a_t^N)$ of skill calls; the first skill a_t^1 is applied to the environment, the state transitions to s_{t+1}, and the planner is re-invoked with the updated observation. This setup encourages the model to reason over the full task while grounding each decision in the observed state after prior actions.

3.1 Planner Overview

At inference, APIVOT generates a *multimodal reasoning trace* and a *plan*. The trace interleaves the model's reasoning about task decomposition, action dependencies, and constraints, with *visual thoughts*. The plan specifies a sequence of actions in the planner action space $\mathcal{A}_{\text{plan}}$, including both discrete primitives and placement parameters.

Multimodal reasoning trace. APIVOT generates a reasoning trace $R := r_1 \oplus \ldots \oplus r_m$, where $\oplus$ denotes concatenation and each r_j is a reasoning step associated with an intermediate subgoal ρ_j. As shown in Figure 2, each step consists of text tokens t_j and may optionally include visual tokens v_j. The text tokens describe the subgoal ρ_j, including the object to interact with, its role in advancing the task, and the relevant semantic or geometric constraints. When visualizing the future scene is

useful for verifying subgoal feasibility, the model generates a span of visual tokens v_j, which we call a *visual thought*. Otherwise, $v_j = \varnothing$. This formulation supports adaptive multimodal reasoning. The model can rely on language for semantic reasoning and introduce visual thoughts only when needed.

Visual thought. A visual thought is instantiated in the trace as a fixed-length span of K *visual tokens*, delimited by special start and end tokens: <|image_start|><|image_pad|>^{K}<|image_end|>. These discrete tokens indicate where the visual thought occurs in the autoregressive sequence, while the decoder hidden states at the K visual tokens encode its visual content. We denote these hidden states as the *latent visual state* $H_j \in \mathbb{R}^{K \times d}$, where d is the VLM's hidden dimension. Since H_j remains in the autoregressive context, subsequent tokens can attend to this predicted future-state representation during downstream reasoning and plan generation.

Action plan. Conditioned on the reasoning trace, the model outputs a plan $(a_1, \ldots, a_n)$, where each $a_i \in \mathcal{A}_{\text{plan}}$ specifies a discrete primitive such as `open`, `pick`, or `place`, along with any required placement parameters. Conceptually, the language components of the trace support discrete action selection, while the visual thoughts encode spatial context for grounding placement parameters.

3.2 Supervision from Reference Outputs

We initialize APIVOT from a pretrained VLM that can process visual inputs and conduct text-based reasoning, but cannot generate or use visual thoughts for planning. To plan effectively, we perform supervised fine-tuning (SFT) on reference outputs derived from successful demonstrations. Successful robot demonstrations specify what actions to perform, but do not directly indicate which planning decisions require visual imagination. Each reference output consists of a multimodal reasoning trace and plan, paired with ground-truth images.

Reference output construction. We construct reference outputs by decomposing each demonstration $\tau \in \mathcal{D}$ into a sequence of subgoals. Consecutive `pick`/`place` skill calls are paired, and each subgoal is annotated with three labels: a constraint type, a modality label, and a purpose. The constraint type captures what kind of reasoning is required, while the modality label specifies whether the reference trace should solve it with text alone or include a visual thought. Concretely, we define three constraint types: a *symbolic precondition*, where the decision is independent of continuous parameters (e.g., opening the fridge); a *current geometric constraint*, where the parameter is restricted by the present scene (e.g., a large bowl in a near-full fridge); and *future geometric feasibility*, where the parameter is locally unconstrained but determines whether later subgoals remain feasible (e.g., placing a small bowl so a larger one can fit later). The latter two involve continuous parameters and include cases where greedy refinement of a high-level plan can fail and downward refinability can break down [20]. We compute the constraint type from geometric metrics over depth and camera extrinsics, including surface areas, object footprints, and free-space margins, and map it to a modality label by thresholding the resulting tightness: symbolic-precondition steps are text-only, and geometric steps receive a visual thought once tightness exceeds the threshold. We additionally tag each subgoal with a purpose, such as advancing the task, satisfying a precondition, or clearing an obstacle.

We then assemble the annotated subgoals into a structured skeleton and prompt a high-capacity language model to expand it into a natural-language reasoning trace. We instruct it to preserve the skeleton's structure while adding connective reasoning about task decomposition, subgoal selection, and progress toward the goal. Conditioning expansion on the subgoals and annotations grounds the trace in the physical constraints relevant to planning rather than generic rationalization, while still matching the VLM's natural reasoning style. For every reasoning step labeled with a visual thought, we insert the discrete visual token sequence after the text component of the step. We also attach a target future RGB observation of the environment after the corresponding subgoal is achieved. This image provides supervision for aligning the visual thought to the ground-truth future state at that point in the trace. Using this procedure, we construct SFT datasets with different modality-labeling strategies to induce different behaviors, such as using visual thoughts at every step or adaptively. See Appendix A.2 for more details and examples of reasoning traces.

Visual alignment. To supervise the VLM's latent visual states, we align them to target features obtained by encoding the corresponding ground-truth images with a frozen vision encoder E. We use the pretrained VLM's own vision encoder to encourage the visual thoughts to lie within the model's visual feature space. Let I_j^{gt} be the reference RGB image for the j-th visual thought. We encode I_j^{gt} with E and average-pool the resulting patch features along the patch dimension into K target

features: $Z_j^{\mathrm{gt}} = \mathrm{AvgPool}\big(E(I_j^{\mathrm{gt}})\big) \in \mathbb{R}^{K \times d_E}$, where d_E is the feature dimension of E. We then project each latent visual state H_j into the encoder feature space using a learned projection head $f_\phi : \mathbb{R}^d \to \mathbb{R}^{d_E}$ to obtain $Z_j = f_\phi(H_j)$. Finally, we apply a cosine similarity loss over the projected latent states to encourage the latent tokens to encode information consistent with reference features: $\mathcal{L}_{\mathrm{vis}} = \frac{1}{|\mathcal{B}_{\mathrm{vis}}|} \sum_{j \in \mathcal{B}_{\mathrm{vis}}} 1 - \cos(Z_j, Z_j^{\mathrm{gt}})$, where $\mathcal{B}_{\mathrm{vis}}$ is the set of visual thoughts in the batch.

Training objective. Across stages, we optimize a training objective with three components: $\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{CE,plan}} + \lambda_{\mathrm{CE,trace}}\mathcal{L}_{\mathrm{CE,trace}} + \lambda_{\mathrm{vis}}\mathcal{L}_{\mathrm{vis}}$, where $\lambda_{\mathrm{CE,trace}}$ and λ_{vis} are hyperparameters that control the weights of the loss components. Here, $\mathcal{L}_{\mathrm{CE,trace}}$ and $\mathcal{L}_{\mathrm{CE,plan}}$ are cross-entropy losses computed over tokens in the multimodal reasoning trace and plan, respectively. The $\mathcal{L}_{\mathrm{CE,trace}}$ loss supervises the structure and content of the reasoning trace, including the syntax of visual thoughts, while the $\mathcal{L}_{\mathrm{CE,plan}}$ trains the model to generate the successful plan conditioned on the reasoning trace. Meanwhile, the visual loss $\mathcal{L}_{\mathrm{vis}}$ aligns the latent visual states with the target scene content.

3.3 Training Curriculum

Our three-stage training curriculum progressively teaches the model how to: 1) *use* visual thoughts as intermediate context for planning, 2) *generate* them internally, and 3) decide *when* they are useful.

Stage 1: Visual thought comprehension. The first stage teaches the model to use visual thoughts to guide planning. To isolate this capability from visual thought generation and modality selection, we train on traces where every reasoning step includes a visual thought, and provide the target latent visual states directly to the model. In particular, we inject the corresponding ground-truth visual features into the decoder input at all visual token positions, providing context for generating subsequent reasoning and actions. As the latent visual states are provided, we disable the visual alignment loss and supervise only the discrete output tokens. In doing so, the model learns to generate the reference reasoning text and effectively ground planning in the provided visual states.

Stage 2: Visual thought generation. The second stage teaches the model to generate the visual thoughts itself. As in Stage 1, every reasoning step includes a visual thought, so that the model learns the mechanics of visual thought generation without having to decide when they are necessary. Instead of injecting the ground truth embeddings, the model must now generate its own latent visual states at the visual placeholder positions. We optimize all loss components by setting $\lambda_{\mathrm{vis}} > 0$, encouraging the model to generate the visual tokens and the latent visual state to encode the target scene context.

Stage 3: Adaptive modality selection. The final stage teaches the model to select the appropriate reasoning modality for each subgoal through SFT on reasoning traces that demonstrate geometrically grounded behavior. Specifically, we train on reference traces in which visual thoughts are included only for subgoals identified by simulator-derived geometric heuristics as requiring geometric precision, such as those that require reasoning about limited free space, collision-sensitive placements, or downstream feasibility dependencies. In this stage, we include all training loss components. The trace CE loss supervises modality selection through the presence or absence of visual thoughts in the reference trace, while the visual alignment and plan CE losses preserve latent visual state generation and downstream planning behaviors, respectively. Together, these losses teach the model to predict when to generate visual thoughts conditioned on its available context, which retains the benefits of visual reasoning and reducing unnecessary reasoning cost.

4 Experiments

Our experiments address two primary questions. First, we evaluate whether APIVOT improves long-horizon planning performance, and further analyze when these gains arise. Second, we understand the extent to which APIVOT demonstrates meaningful adaptive modality selection behavior.

4.1 Experimental Setup

Tasks. We design a suite of long-horizon kitchen tasks that require reasoning jointly about semantic and geometric constraints. The tasks systematically vary semantic goals, geometric constraints, and their compositions, enabling controlled evaluation in settings that demand precise geometric reasoning. The suite consists of three task families: CONTAINMENT, SORTING, and STORING LEFTOVERS. CONTAINMENT involves placing objects into a constrained storage space, requiring

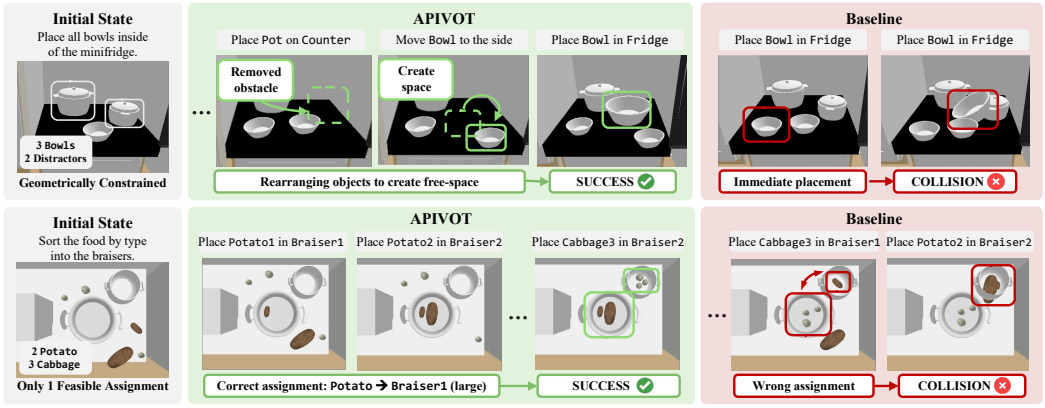


Figure 3: We show execution traces of APIVOT and the top-performing baseline on each task: CONTAINMENT (top) and SORTING (bottom).

reasoning about semantic preconditions and limited free space. SORTING involves assigning objects to containers by type, often under capacity constraints that restrict feasible assignments. STORING LEFTOVERS composes these into a longer-horizon task, with sorting followed by placement into a shared constrained space. We train on CONTAINMENT and SORTING, and hold-out STORING LEFTOVERS to evaluate transfer and compositional generalization. See Appendix A.1 for examples.

We generate all data using the KitchenWorlds [10] simulator. Our pipeline samples diverse task instances across novel scene layouts and object instances, ranging from simple cases with ample free-space to harder ones that require opening doors, removing obstacles, or carefully arranging objects under limited free-space. By abstracting away low-level control, we can directly attribute performance differences to high-level planning rather than errors in sensing or control. For each task instance, we use PDDLStream [21] to generate a successful trajectory and synthesize a corresponding reference output. We train on 2,000 examples per training task and evaluate on datasets for each training task and held-out task, each with 100 examples. Additional dataset details are in Appendix A.2.

We evaluate methods in a receding horizon setup with access to a shared library of primitive actions. Our primary metric is *task success*, defined as the fraction of episodes that succeed. To measure reasoning cost for VLM-based methods, we report *token usage*, computed as the number of tokens in the reasoning trace. For extended thinking VLMs, we include the internal reasoning tokens.

Model implementation. APIVOT is initialized from Qwen3-VL-8B-Instruct [26] and finetuned with LoRA applied to the attention and MLP projections. Full details are provided in Appendix B.

Baselines. We compare APIVOT with three categories of planners: general-purpose VLMs and VLM-based planners, and symbolic planners. For general-purpose VLMs, we use Gemini-3.1-Pro, Gemini-ER-1.5 [27], Qwen3-VL-8B-Instruct (APIVOT's base model), and Qwen3-VL-8B-Thinking. Extended thinking is enabled for Gemini-3.1-Pro, Gemini-ER-1.5, and Qwen3-VL-8B-Thinking. In addition, we choose VLM-TAMP [10] and Reflect-VLM [13] as representative VLM-based planners and FastDownward [23] as the symbolic planner. All planning baselines are finetuned on the same KitchenWorlds data as APIVOT to ensure a fair comparison. More training and implementation details are in Appendix C.

4.2 Task Planning Performance

Overall performance. In Table 1 we compare APIVOT against prior work on held-out instances of the training task families and an unseen, compositional task family. We split prior methods into general-purpose VLM baselines and planning baselines. APIVOT achieves the highest average task success rate of 0.419. Compared to the top-performing VLM baseline, Gemini-ER-1.5, APIVOT demonstrates an 8.1 percentage point improve-

Table 1: Planning performance across tasks.

Model	Avg	CONTAIN	SORT	STORE
VLM Baselines				
Gemini-3.1-Pro	0.245	0.281	0.241	0.213
Gemini-ER-1.5	0.338	0.364	0.339	0.311
Qwen3-VL-8B-Instruct	0.188	0.218	0.183	0.162
Qwen3-VL-8B-Thinking	0.232	0.259	0.234	0.204
Planning Baselines				
FastDownward	0.272	0.331	0.247	0.238
Reflect-VLM	0.292	0.348	0.272	0.257
VLM-TAMP	0.329	0.370	0.311	0.307
APIVOT (Ours)	**0.419**	**0.472**	**0.421**	**0.365**

ment, suggesting that stronger pretrained language-based reasoning is insufficient for these tasks. Compared to the strongest planning baseline, VLM-TAMP, APIVOT improves success by 9.0 points. We hypothesize that these gains come from integrating semantic and geometric reasoning while planning, allowing spatial constraints to shape downstream decisions. These gains are consistent across both training tasks, with APIVOT outperforming prior work by 10.2 points on CONTAINMENT and 8.2 points on SORTING, evaluated on unseen scene configurations.

Notably, APIVOT achieves the highest success rate on STORING LEFTOVERS, a held-out compositional task that combines the two training tasks. This task challenges APIVOT to interpret unseen goals, compose previously learned subtasks into new action sequences, and adapt its modality selection strategy accordingly. APIVOT outperforms the strongest baseline by 5.4 points, sugesting that it can transfer its learned planning strategy to novel compositions of semantic and geometric constraints. Beyond these settings, we find that APIVOT achieves higher success rates in challenging generalization settings involving longer task horizons and increased geometric complexity, while also demonstrating robustness under open-loop execution and partial observability (Appendix D).

Performance under increasing geometric complexity. To understand where these gains arise, Figure 4 breaks down task success as a function of geometric complexity. Since all task families require placing objects into target regions, we measure this using an *occupancy ratio*, defined as the total area of task-relevant objects divided by the available target area.

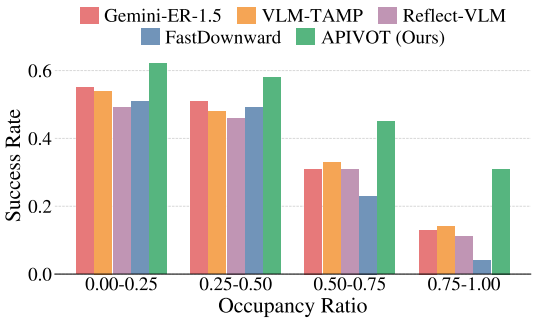


Figure 4: Success rate by *occupancy ratio*.

We compare APIVOT against Gemini-ER-1.5 as the strongest representative VLM baseline, as well as all planning baselines. In low-occupancy settings, tasks are under-constrained and all methods achieve high success. As the occupancy ratio increases, however, tasks require more precise placements, and performance decreases across all approaches. Notably, while baseline performance drops significantly under these conditions, APIVOT remains relatively robust, widening the gap from 7 to 17 percentage points as occupancy increases. Figure 3 illustrates this advantage: in the CONTAINMENT task, APIVOT anticipates that the final bowl cannot be placed without more free space. It therefore rearranges existing objects first, enabling success. In comparison, Gemini-ER-1.5 outputs a symbolically valid plan but places objects directly, causing a collision. See Appendix G for more qualitative examples.

Performance under reasoning budgets. To assess whether visual thoughts encode spatial information more efficiently than language, we compare APIVOT against VLM baselines under matched maximum reasoning-token budgets (Figure 5). APIVOT consistently outperforms all baselines across these settings, with the largest gains at low- and medium- token budgets. While increasing the token allocation improves some reasoning-heavy baselines, their performance remains below that of APIVOT even at much larger scales. This suggests that visual thoughts provide a more token-efficient representation for planning. Instead of relying on long language traces to describe spatial configurations, APIVOT encodes this information directly and compactly with visual thoughts.

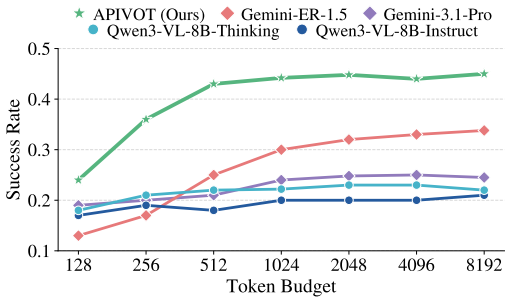


Figure 5: Success rates of VLMs at different token budgets.

Role of visual thoughts. To isolate the contribution of visual thoughts, we train a text-only variant of our method using standard SFT, where the reasoning traces are constructed from the same task instances as APIVOT but use only language. This ablation controls for whether improvements come from domain-specific SFT alone or from the ability to represent intermediate future states visually. Across all tasks, the base model achieves an average success rate of 0.18, while the text-only finetuned variant shows only limited improvement, reaching 0.24 (Appendix F.1). In contrast, APIVOT achieves 0.42, suggesting that while domain-specific SFT is beneficial, the larger gains of APIVOT come

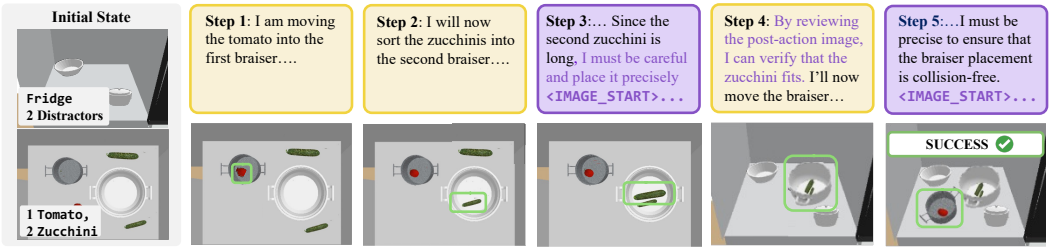


Figure 8: Adaptive reasoning behavior on a held-out STORING LEFTOVERS task. APIVOT reasons in language for semantic decisions such as food-to-container assignment, while selectively generating visual thoughts for geometry-sensitive steps, such as collision-free zucchini placement and final fridge storage. This shows that APIVOT invokes visual reasoning when spatial precision is needed, rather than using visual thoughts uniformly.

from its use of visual thoughts. We provide additional ablations of the training curriculum, visual alignment objective, and geometric supervision strategy in Appendix F.

4.3 Adaptive Modality Selection Behavior

Effective modality selection. To evaluate the benefit of adaptive modality selection, we compare our method (*Adaptive*) against two inference-time ablations: text-only reasoning (*No-Image*) and visual thoughts at every step (*Always-Image*). As shown in Figure 6, *Always-Image* achieves the highest task success but incurs higher token usage from using visual thoughts at every step. In contrast, APIVOT retains 91% of *Always-Image*'s performance while substantially reducing token usage by 39%, placing it at a favorable point on the performance-efficiency frontier. Compared to baselines, APIVOT also lies in the high-success, low-cost region of the tradeoff curve. By using visual thoughts selectively, APIVOT avoids unnecessary reasoning while maintaining their benefits for planning.

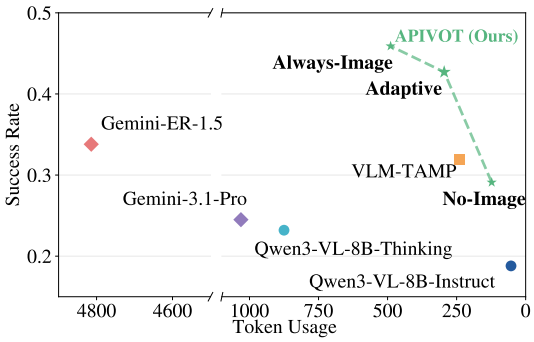


Figure 6: Performance-efficiency trade-off.

Furthermore, we find that APIVOT outperforms both *No-Image* and *Always-Image* under constrained token budgets, demonstrating its advantage in reasoning token efficiency (Appendix E.3).

Geometrically grounded modality selection. Beyond its effect on efficiency, we assess whether APIVOT's adaptive use of visual thoughts reflects meaningful, geometrically-grounded modality selection. We first measure how closely its per-step modality decisions align with the modality labels derived from the geometric heuristics used in training (Appendix 3.2). We find that APIVOT matches these labels with 0.81 accuracy, 0.86 precision, and 0.80 recall, substantially above a rate-matched random baseline that emits visual thoughts at APIVOT's test-time rate (0.51, 0.60, and 0.56, respectively). This suggests that APIVOT does emits visual thoughts based on the geometric state of the scene.

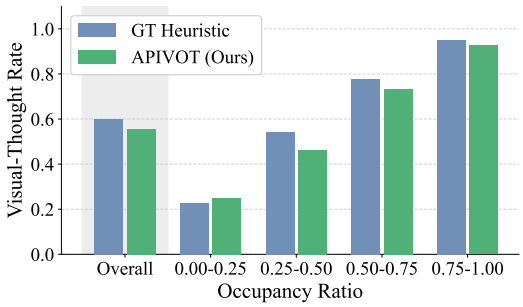


Figure 7: Visual-thought rate by *occupancy ratio*.

Furthermore, we evaluate how often APIVOT uses visual thoughts compared to the ground-truth heuristic. In Figure 7, we report the visual thought rate as geometric complexity increases, measured by the occupancy ratio. We see that APIVOT's usage increases with complexity and closely aligns with the heuristic, using visual thoughts sparingly on unconstrained steps and frequently on the tightest ones. We see an example of such adaptive reasoning in Figure 8, and provide additional

analysis of visual thought content and usage in Appendix E. Together, these results suggest that APIVOT has learned a meaningful geometric proxy for when visual reasoning is useful, rather than applying it uniformly.

5 Conclusion

We propose APIVOT, a VLM-based planner that learns to leverage interleaved language and visual thoughts for long-horizon task planning. Our results show that incorporating visual thoughts improves planning performance, with the largest gains in spatially constrained settings. Beyond improving success, our model learns meaningful modality selection behavior, choosing to reason in language for semantic task structure and in vision when geometric precision is useful. With this adaptive strategy, APIVOT achieves strong performance gains while maintaining efficiency, demonstrating that learning to reason in the right modality is critical to effective multimodal planning.

Limitations. First, our experiments are conducted in the KitchenWorlds simulator. While the task suite captures an ecological range of symbolic and geometric planning challenges, evaluating it in more visually realistic simulators or real-world settings remains an important next step. In practice, APIVOT can be extended by applying the same problem formulation and adapting the data-generation pipeline to real-world environments, with the integration of off-the-shelf perception modules. Second, our learned latent visual representations are trained within our task distribution and may not capture the full variability of real-world geometry and appearance. Scaling visual pretraining to larger, real-world datasets or leveraging advances in unified text-and-image generation models could further improve performance and generalization. Third, adaptive modality selection is learned via supervised finetuning rather than being explicitly optimized for task success. While our model already exhibits semantically meaningful image use, reinforcement learning or test-time adaptation to optimize abstraction selection for task success could yield a more robust policy. Finally, our framework reasons over text and latent images. Extending it to include additional representations (e.g. points or bounding boxes) is a natural direction for enabling it to reason at more levels of abstraction, improving planning performance and efficiency.

Acknowledgments and Disclosure of Funding

This paper is based upon work supported by the Air Force Office of Scientific Research under award numbers FA9550-25-C-B010 and YIP FA9550-23-1-0127, the Stanford Human-Centered AI Institute Hoffman-Yee grant program, ONR MURI N00014-22-1-2740, ONR MURI N00014-24-1-2748, Singapore MDDI AIVP-2025-008, and Analog Devices.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, August 2022. arXiv:2204.01691 [cs].

[2] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents, March 2022. arXiv:2201.07207 [cs].

[3] 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. Inner Monologue: Embodied Reasoning through Planning with Language Models, July 2022. arXiv:2207.05608 [cs].

[4] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency, September 2023. arXiv:2304.11477 [cs].

[5] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Program generation for situated robot task planning using large language models. *Autonomous Robots*, 47(8):999–1012, 2023.

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

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

[8] Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. Embodied task planning with large language models. *arXiv preprint arXiv:2307.01848*, 2023.

[9] Frank Joublin, Antonello Ceravola, Pavel Smirnov, Felix Ocker, Joerg Deigmoeller, Anna Belardinelli, Chao Wang, Stephan Hasler, Daniel Tanneberg, and Michael Gienger. CoPAL: Corrective Planning of Robot Actions with Large Language Models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8664–8670, May 2024.

[10] Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Guiding Long-Horizon Task and Motion Planning with Vision Language Models, October 2024. arXiv:2410.02193 [cs].

[11] Shu Wang, Muzhi Han, Ziyuan Jiao, Zeyu Zhang, Ying Nian Wu, Song-Chun Zhu, and Hangxin Liu. LLM3:Large Language Model-based Task and Motion Planning with Motion Failure Reasoning, August 2024. arXiv:2403.11552 [cs].

[12] Marta Skreta, Zihan Zhou, Jia Lin Yuan, Kourosh Darvish, Alán Aspuru-Guzik, and Animesh Garg. RePLan: Robotic Replanning with Perception and Language Models, February 2024. arXiv:2401.04157 [cs].

[13] Yunhai Feng, Jiaming Han, Zhuoran Yang, Xiangyu Yue, Sergey Levine, and Jianlan Luo. Reflective Planning: Vision-Language Models for Multi-Stage Long-Horizon Robotic Manipulation, February 2025. arXiv:2502.16707 [cs].

[14] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic Control via Embodied Chain-of-Thought Reasoning, March 2025. arXiv:2407.08693 [cs].

[15] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. CoT-VLA: Visual Chain-of-Thought Reasoning for Vision-Language-Action Models, March 2025. arXiv:2503.22020 [cs].

[16] Yucheng Hu, Jianke Zhang, Yuanfei Luo, Yanjiang Guo, Xiaoyu Chen, Xinshu Sun, Kun Feng, Qingzhou Lu, Sheng Chen, Yangang Zhang, Wei Li, and Jianyu Chen. BagelVLA: Enhancing long-horizon manipulation via interleaved vision-language-action generation, February 2026. arXiv:2602.09849 [cs.RO].

[17] Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Dong Wang, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, Maoqing Yao, Haoran Yang, Jiacheng Bao, Bin Zhao, and Xuelong Li. EO-1: An open unified embodied foundation model for general robot control, August 2025. arXiv:2508.21112 [cs.RO].

[18] Yifu Yuan, Haiqin Cui, Yibin Chen, Zibin Dong, Fei Ni, Longxin Kou, Jinyi Liu, Pengyi Li, Yan Zheng, and Jianye Hao. From seeing to doing: Bridging reasoning and decision for robotic manipulation, May 2025. arXiv:2505.08548 [cs.RO].

[19] Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, et al. $\pi_{0.7}$: A steerable generalist robotic foundation model with emergent capabilities, April 2026. arXiv:2604.15483 [cs.LG].

[20] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated Task and Motion Planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):265–293, May 2021.

[21] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning, March 2020. arXiv:1802.08705 [cs].

[22] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins Sri, Anthony Barrett, Dave Christianson, et al. Pddl—the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[23] M. Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26:191–246, July 2006. arXiv:1109.6051 [cs].

[24] Yiqing Xu, Jiayuan Mao, Linfeng Li, Yilun Du, Tomas Lozáno-Pérez, Leslie Pack Kaelbling, and David Hsu. "set it up": Functional object arrangement with compositional generative models. *The International Journal of Robotics Research*, page 02783649251378198, 2024.

[25] Muyang Yan, Miras Mengdibayev, Ardon Floros, Weihang Guo, Lydia E. Kavraki, and Zachary Kingston. Using VLM Reasoning to Constrain Task and Motion Planning, March 2026. arXiv:2510.25548 [cs].

[26] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL Technical Report, November 2025. arXiv:2511.21631 [cs].

[27] Gemini Robotics ER 1.6 Model Card.

[28] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. Emerging properties in unified multimodal pretraining, 2025. arXiv:2505.14683 [cs.CV].

[29] Jinheng Xie, Zhenheng Yang, and Mike Zheng Shou. Show-o2: Improved native unified multimodal models, 2025. arXiv:2506.15564 [cs.CV].

[30] Zhiheng Liu, Weiming Ren, Haozhe Liu, Zijian Zhou, Shoufa Chen, Haonan Qiu, Xiaoke Huang, Zhaochong An, Fanny Yang, Aditya Patel, Viktar Atliha, Tony Ng, Xiao Han, Chuyan Zhu, Chenyang Zhang, Ding Liu, Juan-Manuel Perez-Rua, Sen He, Jürgen Schmidhuber, Wenhu Chen, Ping Luo, Wei Liu, Tao Xiang, Jonas Schult, and Yuren Cong. TUNA: Taming unified visual representations for native unified multimodal models, 2025. arXiv:2512.02014 [cs.CV].

[31] Zhiheng Liu, Weiming Ren, Xiaoke Huang, Shoufa Chen, Tianhong Li, Mengzhao Chen, Yatai Ji, Sen He, Jonas Schult, Belinda Zeng, Tao Xiang, Wenhu Chen, Ping Luo, Luke Zettlemoyer, and Yuren Cong. Tuna-2: Pixel embeddings beat vision encoders for multimodal understanding and generation, April 2026. arXiv:2604.24763 [cs.CV].

[32] Shengbang Tong, David Fan, John Nguyen, Ellis Brown, Gaoyue Zhou, Shengyi Qian, Boyang Zheng, Théophane Vallaeys, Junlin Han, Rob Fergus, Naila Murray, Marjan Ghazvininejad, Mike Lewis, Nicolas Ballas, Amir Bar, Michael Rabbat, Jakob Verbeek, Luke Zettlemoyer,

Koustuv Sinha, Yann LeCun, and Saining Xie. Beyond language modeling: An exploration of multimodal pretraining, March 2026. arXiv:2603.03276 [cs.CV].

[33] Zhaochen Su, Peng Xia, Hangyu Guo, Zhenhua Liu, Yan Ma, Xiaoye Qu, Jiaqi Liu, Yanshu Li, Kaide Zeng, Zhengyuan Yang, Linjie Li, Yu Cheng, Heng Ji, Junxian He, and Yi R. Fung. Thinking with Images for Multimodal Reasoning: Foundations, Methods, and Future Frontiers, July 2025. arXiv:2506.23918 [cs].

[34] Ang Li, Charles Wang, Deqing Fu, Kaiyu Yue, Zikui Cai, Wang Bill Zhu, Ollie Liu, Peng Guo, Willie Neiswanger, Furong Huang, Tom Goldstein, and Micah Goldblum. Zebra-CoT: A dataset for interleaved vision language reasoning, July 2025. arXiv:2507.16746 [cs.CV].

[35] Chengzu Li, Wenshan Wu, Huanyu Zhang, Yan Xia, Shaoguang Mao, Li Dong, Ivan Vulić, and Furu Wei. Imagine while Reasoning in Space: Multimodal Visualization-of-Thought, January 2025. arXiv:2501.07542 [cs].

[36] Yiming Qin, Bomin Wei, Jiaxin Ge, Konstantinos Kallidromitis, Stephanie Fu, Trevor Darrell, and Xudong Wang. Chain-of-Visual-Thought: Teaching VLMs to See and Think Better with Continuous Visual Tokens, November 2025. arXiv:2511.19418 [cs].

[37] Zeyuan Yang, Xueyang Yu, Delin Chen, Maohao Shen, and Chuang Gan. Machine Mental Imagery: Empower Multimodal Reasoning with Latent Visual Tokens, June 2025. arXiv:2506.17218 [cs].

[38] Jiawei Gu, Yunzhuo Hao, Huichen Will Wang, Linjie Li, Michael Qizhe Shieh, Yejin Choi, Ranjay Krishna, and Yu Cheng. ThinkMorph: Emergent Properties in Multimodal Interleaved Chain-of-Thought Reasoning, February 2026. arXiv:2510.27492 [cs].

[39] Zixian Ma, Jianguo Zhang, Zhiwei Liu, Jieyu Zhang, Juntao Tan, Manli Shu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Caiming Xiong, Ranjay Krishna, and Silvio Savarese. LATTE: Learning to Think with Vision Specialists, September 2025. arXiv:2412.05479 [cs].

[40] Haozhe Wang, Alex Su, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel Reasoner: Incentivizing Pixel-Space Reasoning with Curiosity-Driven Reinforcement Learning, October 2025. arXiv:2505.15966 [cs].

[41] Arijit Ray, Ahmed Abdelkader, Chengzhi Mao, Bryan A. Plummer, Kate Saenko, Ranjay Krishna, Leonidas Guibas, and Wen-Sheng Chu. Mull-tokens: Modality-agnostic latent thinking, 2025. arXiv:2512.10941 [cs.CV].

[42] Ruijie Lu, Yiyang Ma, Xiaokang Chen, Lingxiao Luo, Zhiyu Wu, Zizheng Pan, Xingchao Liu, Yutong Lin, Hao Li, Wen Liu, Zhewen Hao, Xi Gao, Shaoheng Nie, Yixuan Wei, Zhenda Xie, Ting Chen, and Gang Zeng. Thinking with visual primitives, 2026. Technical report.

[43] Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. AdaCoT: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning, 2025. arXiv:2505.11896 [cs.LG].

[44] William Chen, Suneel Belkhale, Suvir Mirchandani, Oier Mees, Danny Driess, Karl Pertsch, and Sergey Levine. Training strategies for efficient embodied reasoning, 2025. arXiv:2505.08243 [cs.RO].

[45] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. Large Language Models Can Be Easily Distracted by Irrelevant Context, June 2023. arXiv:2302.00093 [cs].

[46] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the Middle: How Language Models Use Long Contexts, November 2023. arXiv:2307.03172 [cs].

[47] Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. How Easily do Irrelevant Inputs Skew the Responses of Large Language Models?, September 2024. arXiv:2404.03302 [cs].

[48] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[49] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[50] Weiyu Liu, Neil Nie, Ruohan Zhang, Jiayuan Mao, and Jiajun Wu. Learning compositional behaviors from demonstration and language. In *8th Annual Conference on Robot Learning*, 2024.

Supplementary for APIVOT: Adaptive Planning with Interleaved Vision-Language Thoughts

The appendix is organized as follows: In Appendix A, we include details about the task suite and data generation pipeline. In Appendix B, we describe the implementation of our model, including training and compute details. In Appendix C, we provide details about the baseline methods and evaluation protocol. Finally, we provide additional experimental results (Appendix D), analysis of visual thoughts (Appendix E), ablations (Appendix F), and qualitative examples (Appendix G).

A Task Suite and Datasets

This section describes the task suite and data generation pipeline, including scene sampling, reference plan construction, and reasoning trace generation. We then provide examples of the training data. Lastly, we provide the dataset statistics.

A.1 Task Suite

We define three task families: CONTAINMENT, SORTING, and STORING LEFTOVERS, as shown in Figure 9. These families test different combinations of constraints, including semantic preconditions, obstruction handling, geometric placement under limited free space.

A.1.1 CONTAINMENT.

Goal. The goal of this task is to place a set of target objects (e.g. bowls) into a region (e.g. fridge or cabinet). The scene is initialized with all target objects visible on the counters, while the target region may vary in accessibility, obstruction, and available free space.

Core Challenges. The core challenges include recognizing semantic preconditions (e.g. door is open), resolving geometric constraints (e.g. remove the distractor object to make space for the target objects), and planning precisely if geometrically constrained (e.g. precise target object placement). Thus, harder instances require reasoning about accessibility, obstruction, and placement feasibility.

Variations. We vary the number and sizes of target objects, the door state of the container, whether the target region is empty or obstructed, the number of distractor objects, and the amount of available free space. This produces instances ranging from simple open-container cases with ample space to constrained cases requiring prerequisite actions, obstacle removal, and precise placement.

A.1.2 SORTING.

Goal. The goal is to sort a set of food objects by type into separate target containers. Each instance contains two food types (e.g., zucchini and tomato) and two containers (e.g., braisers), and the robot must place all objects of each type into a separate container. The scene is initialized with all target objects and containers visible on the counter, so the core challenge is semantic assignment under capacity constraints.

Core Challenges. A valid assignment of food types to containers depends on the object geometries and container capacities. To verify whether an assignment is valid, we sample placements in the simulator to check whether all objects can be placed collision-free inside their assignment containers. We define two assignment regimes: both-valid and one-valid. In the both-valid regime, either type can fit into either container. In the one-valid regime, one type exceeds the capacity of the smaller container and only one assignment succeeds, requiring a unique mapping from food type to container. Thus, harder instances cannot be solved by semantic grouping alone. They require the planner to perform assignment based on capacity and placement feasibility.

Variations. To induce these challenges, we vary the number and sizes of each object type and the relative capacities of the containers. This produces samples for both regimes that range in how visually obvious the assignment is.

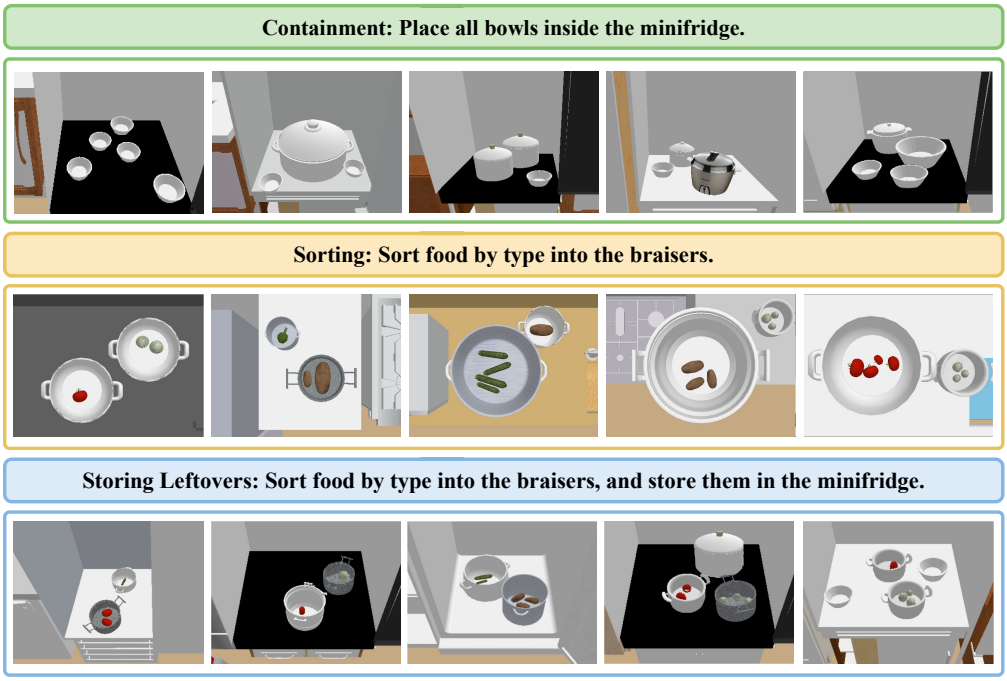


Figure 9: KitchenWorlds task suite with increasing complexity. CONTAINMENT (above) requires placing target objects into constrained storage regions, testing accessibility, obstruction handling, and free-space reasoning. SORTING (middle) requires semantic reasoning over object types to assign food objects to containers, while capacity constraints make some assignments geometrically infeasible. STORING LEFTOVERS (below) composes both challenges into a longer-horizon held-out task, requiring the robot to sort food into containers and then store them in a constrained fridge.

A.1.3 STORING LEFTOVERS.

Goal. The goal is to store a set of food objects as leftovers by first sorting them into containers and then placing the containers in the fridge. Each instance contains two food types, two containers (e.g., braisers), and a minifridge. The robot must place all objects of each food type into a separate container, and then store the filled containers in the fridge. This task combines the capacity-based assignment challenge from SORTING with the access and free-space constraints from CONTAINMENT. We hold out this task family from training to evaluate whether models can generalize from the component families to a novel compositional task.

Core Challenges. A valid plan depends on both the food-to-container assignment and the feasibility of storing the resulting containers in the fridge. As in Sorting, the assignment of food types to containers may depend on object geometries and container capacities. In addition, the final fridge placement may require opening the fridge, clearing obstacles, and arranging the containers within limited free space. Thus, harder instances require reasoning about how early assignment and placement decisions affect downstream storage feasibility.

Variations. As in the SORTING and CONTAINMENT tasks, we vary the number and sizes of food objects, the relative capacities of the containers, the fridge door state, and fridge interior. This produces longer-horizon tasks that require composing semantic grouping, capacity reasoning, and geometric placement constraints.

A.2 Data Generation Pipeline

We generate all training and evaluation data in KitchenWorlds [10], a PyBullet-based simulation environment. The data generation pipeline consists of the following four stages:

Step 1: Scene sampling. Given a task family, we sample a scene that includes the task-relevant object categories. Furthermore, we systematically vary the initial scene layout, including the number of task-relevant objects, distractors, and level of geometric constraint. These variations produce samples with varying levels of difficulty, from simple settings with ample free-space to harder cases requiring opening doors, removing obstacles, or carefully arranging objects under tight space.

Step 2: Successful reference plan generation. To generate a reference plan for a sampled task instance, we use PDDLStream [21] with FastDownward [23] to generate a feasible executable plan, and then convert the resulting feasible executable trajectory into the compact action space predicted by our model.

If a successful trajectory is found, we process the PDDLStream output into a reference plan. The original output contains both high-level actions and low-level continuous parameters, such as object poses, grasps, robot configurations, and motion trajectories. Since our model predicts grounded high-level actions rather than low-level robot trajectories, we filter the PDDLStream plan to retain the actions corresponding to `open`, `pick`, and `place`. Specifically, `open` actions are parameterized by the articulated object, `pick` primitives are parameterized by the object to grasp, and `place` primitives are parameterized by the object being placed, the target support surface, and a normalized image-space placement point obtained from the grounded placement pose. This yields a successful, simulator-validated reference plan consisting of grounded high-level actions.

In addition, we save the simulator state, symbolic plan information, and low-level command sequence. These saved artifacts allow us to replay the plan, compute per-action metadata, and render RGB observations used to construct reference outputs needed for supervised training.

Step 3: Structured skeleton construction. Given the reference plan, our goal is to construct an interleaved language-vision reasoning trace for supervised finetuning. To do so, we first extract a structured skeleton for this trace.

First, we decompose the reference plan into a sequence of subgoals. Concretely, `open` actions become an *open* subgoal, and continuous `pick-place` actions are grouped into a single *place* subgoal. Place subgoals are further categorized by whether the object is being placed into the goal receptacle or moved elsewhere, such as when clearing an obstacle or repositioning an object. This sequence of subgoals captures the structure of the reference plan while abstracting away action-level details.

Next, each subgoal is annotated with a purpose, constraint type, and modality label. Possible subgoal purposes include advancing the task, satisfying semantic precondition, and clearing away obstacles. To determine the constraint type and modality label, we use simulator-derived metadata obtained by replaying the saved plan and saving geometric metrics. For place subgoals, we compute two geometric heuristics: 1) *free space ratio*: the fraction of the target surface area that remains free after the object is placed, and 2) *future feasibility margin*: the free space remaining after the current placement, relative to the total footprint of remaining goal objects. A low free space ratio indicates that the object nearly fills the available space and requires precise placement, while a low future feasibility margin indicates that the current object fits but its placement must be precise to preserve space for future objects.

These heuristics are computed from object bounding boxes projected onto the X-Y plane. Although these metrics are approximate, they provide a consistent signal for identifying subgoals that depend on spatial constraints. In practice, we obtain the bounding boxes directly from the simulator, but we can also use off-the-shelf segmentation models when ground-truth information is unavailable.

Based on these annotations, each subgoal is labeled with a constraint type: symbolic precondition, current geometric constraint, or future geometric feasibility. We use these constraint annotations to assign a modality label to each reasoning step. Steps with a symbolic precondition constraint are assigned text-only reasoning. For other steps, we assign a visual thought if either one of the two geometric heuristics falls below a predefined threshold.

Step 4: Reference trace expansion. After constructing the annotated skeleton trace, we use a high-capacity language model to expand it into a natural-language reasoning trace. The skeleton specifies the sequence of decision points, the corresponding actions, each decision's purpose, its primary planning constraint, the assigned modality, and any associated visual reference. The LLM is instructed to preserve this structure while adding connective reasoning about task decomposition, subgoal selection, progress toward the goal, relevant physical constraints, and why language alone or a visual thought is appropriate for each step. The generated reasoning is then parsed back into a structured format, so that the final reference output contains both the natural-language reasoning trace and the simulator-validated target plan. We use `gemini-3-flash-preview` for this step. The trace expansion prompt is provided in Appendix H.1.

For each reasoning step assigned a visual modality, we insert a visual thought after the text component of the step. A visual thought is represented in the trace by a discrete visual-token span `<|image_start|><|image_pad|>...<|image_pad|><|image_end|>`, and is paired with a rendered RGB observation after the subgoal has been achieved. During training, the rendered image provides the latent target for the visual thought tokens.

A.3 Dataset Examples

In Figure 10 and Figure 11, we provide dataset examples for each training task family. These demonstrate the task instruction and input scene, alongside the reference outputs (reasoning trace, subgoal images, and plan).

A.4 Dataset Statistics

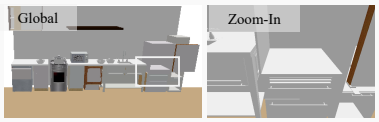
Using the above pipeline, we construct datasets for each training stage and evaluation. For each training stage, the dataset contains 2,000 examples from each training task families. CONTAINMENT instances contain 1–4 target objects, and SORTING instances contain 2–5 target objects. We also systematically vary the presence of distractor objects, scene layouts, and initial states.

For evaluation, each dataset contains 100 examples. We include two in-distribution (ID) datasets, one for each training task family, sampled from the same distribution of target objects and constraints as the training data. We also evaluate on the held-out compositional task family, STORING LEFTOVERS, with instances containing 2-5 target objects for sorting. Finally, we construct two within-family out-of-distribution (OOD) datasets to evaluate generalization to longer task horizons and increased task complexity. For task horizon, we increase it to 5–7 target objects for CONTAINMENT and 6–10 for SORTING. For complexity, we increase the number of prerequisite actions required, defined based on actions that do not directly contribute to the goal (e.g. removing obstacles, opening doors).

Containment: Store the bowls inside the minifridge.

Task: Store the bowls inside the minifridge.

Initial State



Reference Plan

pick(bowl#3_28)
place(bowl#3_28, minifridge_10, [576, 440])
pick(bowl#1_25)
place(bowl#1_25, minifridge_10, [544, 479])
pick(bowl#2_26)
place(bowl#2_26, minifridge_10, [552, 414])

Reference Reasoning Trace

Step 1: There are three large bowls visible on the kitchen counters, and the minifridge is initially open and empty inside. To complete the task, I just need to place the bowls into the minifridge. I will start by placing bowl#3_28. Since there is ample free space, text is sufficient for this step.

Step 2: : I will now place bowl#1_25 into the minifridge to continue fulfilling the task requirements. My main concern is ensuring that I leave enough room for the last bowl that needs to be stored inside later. I will use a post-action image to verify that this bowl's position preserves sufficient space for the remaining objects. The image representation is `<|image_start|>...<|image_end|>.`

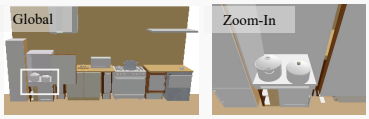
Step 3: Finally, I will place bowl#2_26 inside the minifridge to complete the task. Because the fridge is now crowded, I must be extremely careful to avoid collisions with the objects already inside. A post-action image is necessary to confirm that the placement is precise and fits within the tight geometric constraints. The image representation is `<|image_start|>...<|image_end|>.`

Subgoal Images



Task: Store the bowls inside the minifridge.

Initial State



Reference Plan

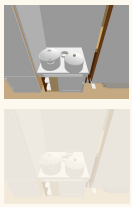
pick(bowl#1_32)
place(bowl#1_32, minifridge_14, [192, 524])
pick(bowl#2_33)
place(bowl#2_33, minifridge_14, [211, 513])

Reference Reasoning Trace

Step 1: There are two bowls in this kitchen environment. The minifridge already contains two large pots, but I think there is still enough remaining space because the bowls are relatively small. I will first place bowl#1_32 in the open area that avoids colliding with the pots and leaves space for the second bowl. Since I want to be careful, I will use a post-action image to verify that the bowl fits cleanly. The image representation is `<|image_start|>...<|image_end|>.`

Step 2: I will now place bowl#2_33 inside the fridge to complete the task. The second bowl is also small, so there is still sufficient space in the fridge to the right of the first bowl. I do not need to worry about constraining future placements, so text is sufficient for this step.

Subgoal Images



Task: Store the bowls inside the minifridge.

Initial State



Reference Plan

pick(kitchenpot#1_29)
place(kitchenpot#1_29, sink_counter_right_6, [392, 473])
pick(bowl#2_16)
place(bowl# 2_16, minifridge_11, [751, 578])
pick(bowl#3_17)
place(bowl#3_17, minifridge_11, [729, 571])
pick(bowl#1_15)
place(bowl#1_15, minifridge_11, [743, 553])

Reference Reasoning Trace

Step 1: I must place three bowls inside the fridge. There are two large pots inside of the fridge right now. To create space, I need to remove kitchenpot#1 from the fridge to make more room for the primary task objects. I will use the left sink counter as the destination surface. It is quite large, so there is no need to verify the placement with a post-action image.

Step 2: I will now place bowl#2_16 into the fridge to begin fulfilling the task requirements. There is plenty of open space now, but I want to be careful so that I preserve enough space for the later placements. I will use a post-action image to ensure that this placement is precise. The image representation is `<|image_start|>...<|image_end|>.`

Step 3: I will proceed by placing bowl#3_17 inside the fridge. The fridge is still pretty empty. Since only one bowl remains to be placed after this step, the current placement is unlikely to create significant downstream constraints. Therefore, I can proceed with a reasonable placement using text reasoning alone.

Step 4: Finally, I am moving bowl#1_15 into the minifridge to complete the set of goal objects. The interior is now very full, making it critical that this last bowl is placed perfectly to fit without hitting anything else. I will check a post-action image to ensure the final placement is successful and collision-free. The image representation is `<|image_start|>...<|image_end|>.` The task is now complete, and I will output the final answer.

Subgoal Images

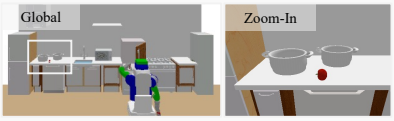


Figure 10: Examples for CONTAINMENT, showing the inputs and reference outputs.

Sorting: Sort the food by type into the braisers.

Task: Sort the tomatoes and zucchinis by type into the braisers.

Initial State



Reference Plan

pick(veggietomato_28)
place(veggietomato_28, braiserbody#1_26, [364, 478])
pick(veggietomato#1_29)
place(veggietomato#1_29, braiserbody#1_26, [363, 491])
pick(veggiezucchini_30)
place(veggiezucchini_30, braiserbody#2_27, [311, 442])

Reference Reasoning Trace

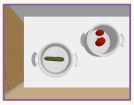
Subgoal Images

Step 1: I will begin by sorting the two tomatoes. I will place veggietomato_28 into braiserbody#1_26 as part of the sorting task. Since the braiser is initially empty, there is plenty of open space. The tomato can be placed freely without precise positioning, so text descriptions provide sufficient information for a successful placement.

Step 2: Next, I'll move the second tomato, veggietomato#1_29, into the same braiser. Since the first tomato is already inside, I need to be more careful about the placement to ensure the two tomatoes fit without colliding. I will examine a post-action image to verify that the second tomato is placed correctly within the remaining space. The image representation is <|image_start|>...<|image_end|>.

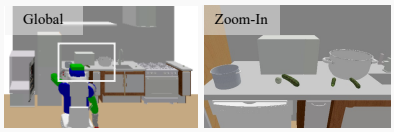


Step 3: Finally, I will place veggiezucchini_30 into braiserbody#2_27 to complete the task. Because the zucchini is long, I need to position it carefully so that it fits properly without hitting the sides of the braiser. I will use a post-action image to confirm that the zucchini successfully fits inside the braiser. The image representation is <|image_start|>...<|image_end|>. The sorting task is now complete, so I will output the final plan.



Task: Sort the cabbages and zucchinis by type into the braisers.

Initial State



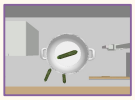
Reference Plan

pick(veggiezucchini_29)
place(veggiezucchini_29, braiserbody#16_45, [575, 486])
pick(veggiezucchini#1_30)
place(veggiezucchini#1_30, braiserbody#16_45, [571, 499])
pick(veggiezucchini#2_31)
place(veggiezucchini#2_31, braiserbody#16_45, [574, 440])
pick(veggiecabbage#1_27)
place(veggiecabbage#1_27, braiserbody#25_54, [433, 471])
pick(veggiecabbage_26)
place(veggiecabbage_26, braiserbody#25_54, [439, 460])
pick(veggiecabbage#2_28)
place(veggiecabbage#2_28, braiserbody#25_54, [445, 476])

Reference Reasoning Trace

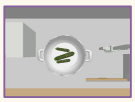
Subgoal Images

Step 1: There are three zucchinis and three cabbages in the scene. Since the zucchinis are longer, I'll sort them into the larger braiser (braiserbody#16_45), starting with the largest zucchini (veggiezucchini_29). To ensure that veggiezucchini_29 fits properly inside the braiser, I need a post-action image to verify its placement. The image representation is <|image_start|>...<|image_end|>.



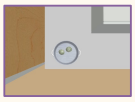
Step 2: Next, I need to place veggiezucchini#1_30 into the same braiser to continue the sorting process. There is currently plenty of room in the container, and this zucchini is shorter. Since I don't need to worry about tight spatial constraints for this specific move, text-based instructions are sufficient for this step.

Step 3: I am moving the last zucchini into the braiser to complete this part of the sorting task. The container still has ample room, but I need to find a precise spot that avoids hitting other items. A post-action image is useful to check veggiezucchini#2_31's placement. The image representation is <|image_start|>...<|image_end|>.



Step 4: Now, I will sort the cabbages into braiserbody#25_54, which is the smaller braiser. I will start by moving veggiecabbage#1_27. The braiser is currently empty, so the placement is not restricted. I can rely on text alone to guide this action without needing visual confirmation of the result.

Step 5: I need to place the next cabbage into the second braiser to keep the food items organized. When placing veggiecabbage_26, I need to be careful to ensure that it doesn't collide with the existing cabbage. A post-action image is necessary to validate that the cabbage is positioned safely within the container. The image representation is <|image_start|>...<|image_end|>.



Step 6: I will move the final cabbage into the second braiser to finish the task. Since the container is very crowded, I must be careful with the placement to ensure a collision-free fit. I'll check a post-action image to verify the final arrangement is successful. The image representation is <|image_start|>...<|image_end|>. The task is now complete, and I will output the final answer.

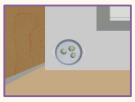


Figure 11: Examples for SORTING, showing the inputs and reference outputs.

B Model and Training Details

We provide implementation details for APIVOT, including the architecture used to represent latent visual thoughts and the training setup used for finetuning.

B.1 Architecture and Visual Thought Implementation

Our model is initialized from Qwen3-VL-8B-Instruct and extends the base VLM decoder to support latent visual thoughts within the autoregressive reasoning trace. We represent each visual thought as a fixed-length span of special tokens: <|image_start|> <|image_pad|>^{K} <|image_end|>, where $K = 16$. Since Qwen3-VL-8B has hidden dimension $d = 4096$, each visual thought corresponds to a latent visual state $H_j \in \mathbb{R}^{16\times 4096}$.

The start and end tokens of the visual-thought span supervise the model to produce valid visual-thought structure, while the hidden states corresponding to the image-pad tokens represent the model's latent visual thought.

B.2 Training and Compute

We finetune APIVOT with LoRA applied to the language-model attention and MLP projection modules, with rank $r = 16$ and scaling $\alpha = 32$.

We train our model with the three-stage supervised fine-tuning curriculum described in Section 3.3. We train on 2000 examples for each task family, for 1 epoch. Across all training stages, we use a per-device batch size 4 with gradient accumulation over 16 steps, optimizer AdamW, cosine learning-rate decay, learning rate 4×10^{-5}, and a warmup ratio 0.08.

For the training objective: In Stage 1, we set $\lambda_{\mathrm{CE,trace}} = 1.0$ and $\lambda_{\mathrm{vis}} = 0$. In Stages 2 and 3, we set $\lambda_{\mathrm{CE,trace}} = 1.0$. and $\lambda_{\mathrm{vis}} = 5.0$.

Training is performed on a single RTX PRO 6000 GPU with 96GB of memory, with each run taking approximately 14 hours.

C Evaluation: Execution Protocol and Baselines

In this section, we describe the closed-loop execution protocol used to evaluate all models, followed by details of the general-purpose VLM and planning baselines.

C.1 Closed-Loop Execution Protocol

We evaluate all methods in the same receding-horizon execution setup, using our simulator.

Formally, the environment is modeled as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T})$, where $\mathcal{S}$ is the state space, $\mathcal{A}_{\text{sim}}$ is the simulator action space, and $\mathcal{T} : \mathcal{S} \times \mathcal{A}_{\text{sim}} \to \mathcal{S}$ is the transition function. In our setup, $\mathcal{A}_{\text{sim}}$ consists of parameterized action primitives (e.g., `open`, `pick`, and `place`) whose continuous parameters (e.g., grasp pose, object pose) are defined in the world frame.

The planner does not directly output actions in $\mathcal{A}_{\text{sim}}$. Instead, it predicts actions in an analogous action space $\mathcal{A}_{\text{plan}}$ that preserves the same discrete primitives but only expresses placement parameters. Specifically, as described in Section 3, the planner action space $\mathcal{A}_{\text{plan}}$ consists of three high-level primitives: `open(obj)`, `pick(obj)`, and `place(obj, target, u, v)`. The `open` action opens articulated containers such as cabinets or refrigerators. The `pick` action selects an object instance, and `place` places the held object on or inside a target object using an image-space placement coordinate (u, v) normalized to the $[0, 1000]$ range.

During execution, planner actions are mapped into $\mathcal{A}_{\text{sim}}$ by lifting any image-space coordinates into world-frame using the simulator state, known camera pose, and scene geometry. Concretely, we unproject the predicted image coordinate into the scene, raycast against the simulator geometry, and place the object at the resulting 3D point. The simulator then applies the corresponding primitive and updates the state according to $\mathcal{T}$.

Episodes terminate on task success, or when the maximum number of execution steps or wall-clock time is reached. Task success is defined as satisfying all goal predicates at termination.

C.2 General-Purpose VLM Baselines

We evaluate all general-purpose VLMs zero-shot to ensure a consistent comparison of their inherent planning capabilities. Concretely, we prompt them directly to produce plans in the same action format as our model. Each prompt contains the current image observation, the task goal, the object list with image-space object locations, and the action schema (See Figure 24). The model outputs a plan, which is parsed and executed using the shared closed-loop protocol above.

For Gemini-based baselines, we evaluate Gemini-3.1-Pro and Gemini-ER-1.5 using greedy decoding with temperature 0. For Qwen baselines, we evaluate Qwen3-VL-8B-Instruct and Qwen3-VL-8B-Thinking. Extended thinking is enabled for Gemini-3.1-Pro, Gemini-ER-1.5, and Qwen3-VL-8B-Thinking. Their thinking tokens are included in token usage measurements.

Separately, we control for the effect of task-specific finetuning through a text-only SFT ablation (Appendix F.1).

C.3 Planning Baselines

We evaluate on specialized planning baselines. For each, we train all learnable components following the task-specific supervision required by its original formulation. We construct this supervision from the same KitchenWorlds data used by APIVOT for a fair comparison.

FastDownward. We evaluate FastDownward [23] as a symbolic planning baseline. Since FastDownward cannot operate directly on RGB observations and instead requires state predicates, we train a predicate prediction model (e.g. BLADE [50]) on our KitchenWorlds dataset to infer the symbolic predicates from the image and language inputs. The inferred state is converted into a PDDL problem and solved with FastDownward using A* search with the LM-cut heuristic. Because FastDownward produces symbolic actions rather than image-space placement coordinates, we ground its placement actions with random sampling.

Reflect-VLM. We evaluate Reflect-VLM [13] as a VLM planning baseline that uses external, imagined future observations to critique and revise its plans before execution. To adapt Reflect-VLM to our task setting, we train both its diffusion model and VLM components on our generated planning dataset using the same KitchenWorlds task instances and action space as our method. At inference, Reflect-VLM receives the current image, goal, and object locations, and proposes a plan with $n = 5$ high-level actions, following their implementation. The proposed plan is rolled out using a diffusion-based dynamics model for a fixed number of imagined steps, producing future observations that are given to the VLM for reflection.

VLM-TAMP. VLM-TAMP [10] is a representative VLM-based task-and-motion planning framework that uses a VLM to propose symbolic planning structure and a downstream planner to produce executable actions. In this framework, the VLM receives the same input as the general-purpose VLM baselines, namely the current image, goal, object locations, and action schema, and is prompted to propose a sequence of formal subgoals. These subgoals are then solved by a symbolic planner and grounded using the same execution interface. In practice, we use Gemini-3.1-Pro as the VLM and train a predicate prediction model on KitchenWorlds data.

D Additional Results

In this section, we provide additional results under out-of-distribution settings, open-loop execution, and partial observability, as well as a failure analysis.

D.1 Out-of-Distribution Generalization

Experiment Setup. We further evaluate APIVOT on out-of-distribution (OOD) settings that increase task horizon and geometric complexity relative to the training distribution. The LONGER HORIZON setting increases the number of target objects, while the INCREASED COMPLEXITY setting includes tighter free-space constraints, and additional prerequisite steps such as opening storage regions or removing obstacles before placement.

Results. As shown in Table 2, APIVOT remains competitive across these harder settings, achieving comparable or stronger performance than the general-purpose VLMs and planning baselines. This suggests that its learned use of visual thoughts can transfer to more complex scenes, although the performance margin is smaller than in-distribution (ID).

However, APIVOT shows one of the largest ID-OOD drops compared to baselines. The OOD settings contain more objects than those seen in training, requiring visual thoughts to encode denser spatial configurations. Thus, we hypothesize that this drop in performance is partly due to limited supervision for generating visual thoughts of more cluttered scenes.

Table 2: Within-family OOD generalization. We report the success rate and relative ID-OOD gap.

	Success Rate ↑		Relative Gap ↓	
Model	Long Horizon	More Complex	Long Horizon	More Complex
VLM Baselines				
Gemini-3.1-Pro	0.225	0.224	0.082	0.086
Gemini-ER-1.5	0.331	0.303	**0.021**	0.104
Qwen3-VL-8B-Instruct	0.177	0.188	0.059	0.000
Qwen3-VL-8B-Thinking	0.199	0.235	0.142	**-0.013**
Planning Baselines				
FastDownward	0.213	0.243	0.217	0.107
Reflect-VLM	0.242	0.252	0.171	0.137
VLM-TAMP	0.281	0.269	0.146	0.182
APIVOT (Ours)	**0.338**	**0.336**	0.193	0.199

D.2 Open-Loop Planning

Experiment Setup. We additionally evaluate all methods under open-loop execution, in which the model commits to a full action sequence without intermediate feedback or replanning. To isolate planning ability from partial observability, we conduct this evaluation on scenes in which all task-relevant objects are initially visible.

Results. We report the success rates in Figure 12. We see that APIVOT achieves the highest open-loop success rate of 0.348 and the smallest relative drop (16.9%) from closed-loop execution. Notably, its advantage over the strongest baseline increases from 8.1 to 11.2 points under open-loop execution. Since open-loop execution requires reasoning about how the scene evolves across multiple actions without intermediate feedback, we attribute these gains to APIVOT's use of visual thoughts, which enable more accurate reasoning about future scene states. By contrast, the larger performance drops for other VLM-based planning approaches, such as Reflect-VLM and VLM-TAMP, suggest that they benefit more from external feedback and replanning.

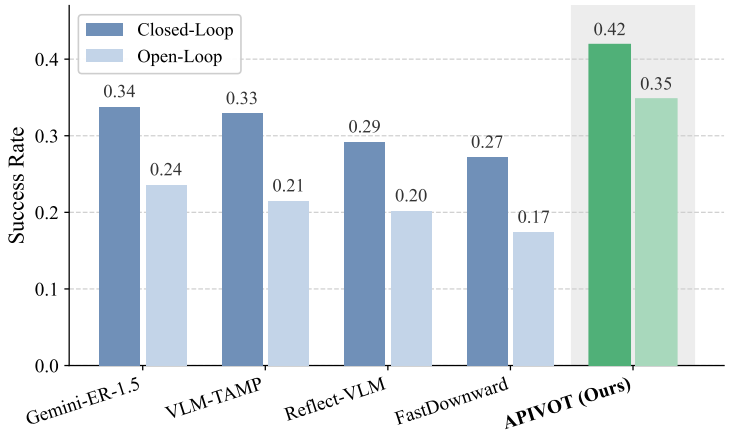


Figure 12: Closed-loop vs. open-loop success rate. APIVOT attains the highest open-loop success and the smallest drop relative to closed-loop execution.

D.3 Behavior under Partial Observability

Experiment Setup. We further analyze APIVOT's performance on the CONTAINMENT task, which features task instances that are initially partially observable as the initial state of the fridge door can be closed. Concretely, we classify samples by the initial state of the door and compare average success rates, token usage, and modality usage of the initial state's reasoning trace.

Results. In Table 3, we report the average success rates and token counts based on the initial state. We see that success is comparable across initially-open and initially-closed samples, suggesting that APIVOT reliably identifies the closed door and resolves the semantic prerequisite. Notably, APIVOT uses fewer tokens when the door is closed, suggesting that it refrains from extended reasoning when the relevant geometric information is not visible.

Table 3: Performance on CONTAINMENT by the initial state of the fridge door.

Initial State	Success Rate	Initial Token Count
Open	0.484	148
Closed	0.459	103

Overall, APIVOT maintains comparable success across observability conditions, while using fewer tokens when the fridge interior is not visible. This suggests that it avoids expending visual thoughts when the relevant geometric information is unavailable, while using them once the interior is observable and visual imagination can be properly grounded.

D.4 Failure Analysis

We classify APIVOT failures based on executed actions into the following primary failure modes:

Imprecise geometric grounding: The high-level symbolic plan is correct, but the predicted spatial parameters are infeasible. We identify these failures when the plan succeeds with oracle simulator placement but fails using the model's predicted points. These failures are more common on tightly constrained scenes, where the tolerance for placement is smaller.

Incorrect action ordering: The plan contains all necessary actions but orders them incorrectly. APIVOT occasionally identifies the relevant objects and operations, but does not correctly model their temporal dependencies. For example, it may plan to clear an obstacle after first placing objects around it, requiring reorganization.

Missing prerequisite actions: APIVOT sometimes omits actions that are necessary to make the task solvable, yet not directly relevant to the goal. APIVOT handles semantic prerequisites (e.g. opening

a closed door) reliably, but still struggles on geometric constraints that require removing an obstacle or rearranging objects that block target locations, where the need for the action is apparent only from the spatial configuration of the scene.

Premature goal completion: Since APIVOT infers task progress from the current observation and object inventory, it sometimes misjudges a partially completed task as finished and ends the episode while the goal remains unsatisfied. We observe that this failure mode is more common on long-horizon tasks, suggesting the need for more explicit progress tracking.

E Analysis of Visual Thoughts and Learned Modality Selection

In this section, we provide additional analysis of the content of visual thoughts (Appendix E.1) and modality selection behavior (Appendix E.2). We also demonstrate the advantage of learned modality selection under different reasoning budgets (Appendix E.3).

E.1 Probing the Content of Visual Thoughts

Experiment Setup. To evaluate whether APIVOT's visual thoughts encode task-relevant geometric information, we train linear probes on top of them to predict three scene attributes: 1) the remaining free space (discretized), 2) whether the planned object will fit, and 3) the number of objects in the target region. We also add a control attribute, the target object category, which is fully specified by the language context. This enables us to isolate whether APIVOT's advantage reflects a general improvement in representation quality or is specific to geometry.

We compare APIVOT's latent visual thoughts against two baselines, the text-only hidden states and randomly initialized vectors, as well as a chance-level baseline.

Results. We report balanced accuracy across all targets in Figure 13. APIVOT outperforms the text-only baseline by 38 points on free-space prediction, 22 points on object-fit prediction, and 18 points on object count, with additional gains over the random-vector and chance baselines. These results directly show that the visual thoughts meaningfully encode scene-specific geometric information. Importantly, APIVOT outperforms the text-only baseline by only 3 points on the language-specified control (target object category), while achieving much larger gains on the geometric probes. This indicates that APIVOT preserves strong language context, while its primary advantage is specific to geometric content.

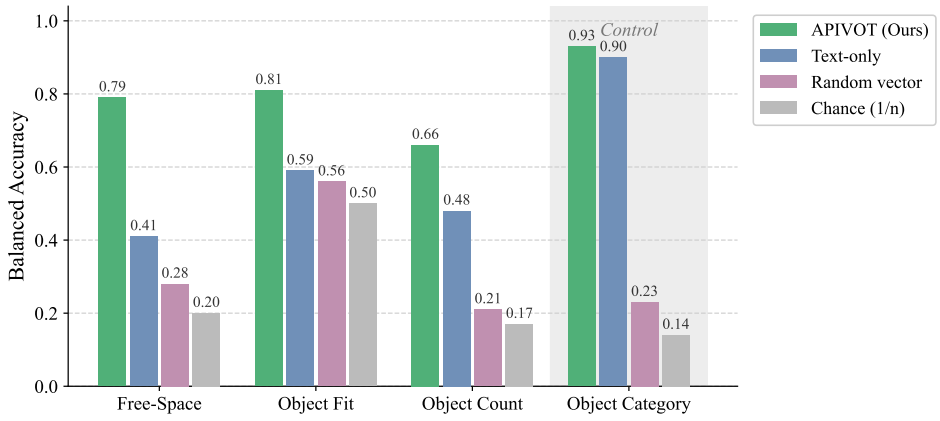


Figure 13: Balanced accuracy of linear probes trained on APIVOT's visual thoughts and baseline representations. APIVOT dominates on the geometric attributes but only marginally exceeds the text-only baseline on the language-specified control (object category).

E.2 Alignment with geometric heuristics

To quantify how well APIVOT's learned modality selection matches our geometric heuristics, we compare its per-step modality decisions against the heuristic-derived labels and report accuracy, precision, and recall. As a baseline, we use a rate-matched random predictor that independently emits a visual thought with probability equal to APIVOT's test-time visual-thought rate (0.56). As shown in Table 4, APIVOT substantially outperforms the random baseline across all three metrics, confirming that its modality decisions align with the geometric state of the scene rather than being emitted at a fixed rate.

Table 4: Alignment of APIVOT's modality selection with the geometric-heuristic labels, measured by accuracy, precision, and recall. Random is a rate-matched baseline that emits visual thoughts at APIVOT's test-time rate (0.56).

Method	Accuracy	Precision	Recall
APIVOT (Ours)	**0.805**	**0.862**	**0.802**
Random (rate-matched)	0.511	0.599	0.557

E.3 Benefits of Modality Selection under Matched Reasoning Budgets

Experiment Setup. In order to demonstrate the benefit of adaptivity in improving reasoning efficiency, we evaluate all three inference-time modes of APIVOT (Adaptive, Always-Image, No-Image) under matched maximum reasoning budgets of 128, 256, and 512 tokens. We additionally compare against a baseline that randomly generates visual thoughts at the same rate as Adaptive.

Results. In Figure 14, we report the success rates of each inference mode under different token budgets. Adaptive achieves the highest success at the lower token budgets (128 and 256), and only performs below Always-Image at the highest budget (512). In low-budget settings, Always-Image often exhausts the available tokens before completing planning, causing its performance to drop substantially, even below No-Image. By contrast, Adaptive allocates visual reasoning selectively, achieving strong performance even under tight budgets. It also consistently outperforms the random baseline, demonstrating the importance of deciding *when* to use visual reasoning rather than using visual thoughts at a fixed rate.

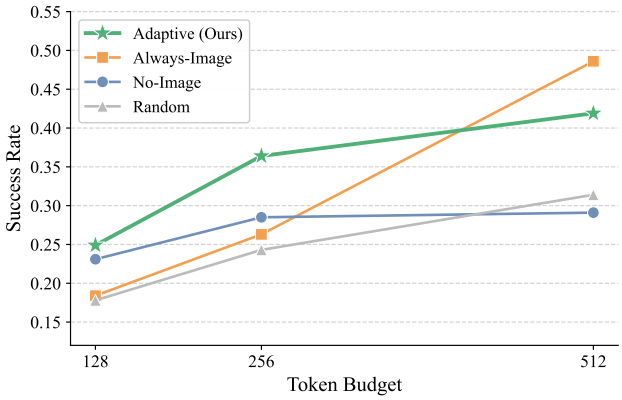


Figure 14: Performance of difference inference-time modalities under matched token budgets.

F Ablations

We provide additional ablations of the main components of APIVOT's training and reasoning pipeline. Concretely, we analyze the role of domain-specific fine-tuning (Appendix F.1), the training and inference reasoning modality (Appendix F.2), the training curriculum (Appendix F.3), the visual-alignment objective (Appendix F.4), the placement and frequency of modality-label supervision (Appendix F.5), and sensitivity to the geometric-heuristic thresholds (Appendix F.6).

F.1 Text-Only Fine-Tuning

Experiment Setup. To isolate whether APIVOT's gains come from domain-specific finetuning alone, we train a text-only variant using the same Qwen3-VL-8B-Instruct backbone, task instances, and expert plans as APIVOT. This variant is trained with standard supervised finetuning on the text reasoning trace and final plan, providing a control for whether the model can recover the same planning improvements simply by learning task-specific language rationales and action formats.

Results. As shown in Figure 15, text-only SFT improves over the base Qwen3-VL-8B-Instruct model, increasing average task success from 0.188 to 0.24. This indicates that domain-specific finetuning is useful. The model benefits from exposure to the task distribution, structured reasoning format, and expert action sequences. However, text-only SFT yields limited improvement in comparison to APIVOT, which achieves an average success rate of 0.419. Thus, text-only SFT accounts for only a small fraction of the overall gain. The remaining gap suggests that APIVOT's improvement is not simply due to better task-specific language supervision, but to its ability to represent intermediate future states visually and leverage them for planning.

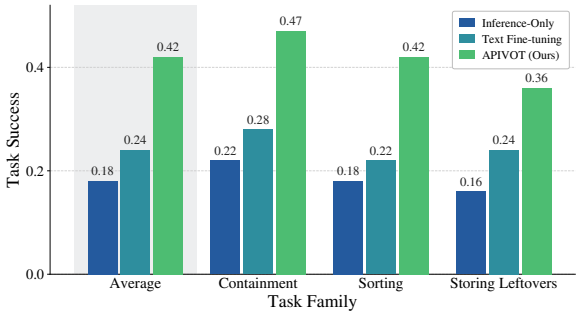


Figure 15: Success rates across all task families.

F.2 Training and Inference Modality

Experiment Setup. To disentangle the effects of training-time multimodal supervision and inference-time multimodal generation, we ablate the modality used at training and inference. We compare APIVOT against a text-only variant finetuned on text-only traces, as in Appendix F.1. For APIVOT, we further ablate the reasoning modality at inference-time: 1) text-only, where visual thoughts are not used, 2) always-image, where images are generated at every step, 3) adaptive, our learned policy that balances performance and cost, and 4) GT oracle, where APIVOT conditions on ground-truth visual thoughts as an upper bound.

Results. As shown in Table 5, image-text training improves planning performance over text-only finetuning overall. Even when inference time reasoning is text-only, the image-text model outperforms the text-only model by 3.8 percentage points. This suggests that multimodal supervision provides benefits beyond explicit visual generation at test time, likely by encouraging planning representations that are more sensitive to geometry.

Inference-time visual thoughts provide an even larger gain, as demonstrated by the performance of the image-always and adaptive ablations. Finally, oracle visual thoughts achieve the highest average success rate, outperforming generated visual thoughts across all task families. This gap suggests that improving the quality of generated visual thoughts could yield additional gains.

Table 5: Effect of training and inference modality on planning performance.

Training	Inference	Avg	CONTAIN	SORT	STORE
Text	Text	0.244	0.281	0.217	0.235
Text+Image (APIVOT)	Text	0.282	0.332	0.267	0.246
	Image (always)	0.459	0.483	0.466	0.428
	Image (adaptive)	**0.419**	**0.472**	**0.421**	**0.365**
	Image (GT, oracle)	0.482	0.494	0.482	0.470

F.3 Training Curriculum

Experiment Setup. APIVOT's three-stage curriculum is designed to progressively build toward learning adaptive modality selection. To test whether the preceding stages are necessary for learning this behavior, we compare training directly from Stage 3 against the full curriculum, as well as variants that add Stage 1 or Stage 2 individually before Stage 3.

Results. We report average success rates in Table 6. Training directly on Stage 3 yields only 0.273 success, 14.6 points below the full curriculum. This demonstrates the difficulty of simultaneously learning to use, generate, and adaptively select visual thoughts. Adding Stage 1 or Stage 2 individually before Stage 3 improves success to 0.345 and 0.301, respectively, showing that each stage provides a meaningful prior for learning adaptive modality selection. Notably, the full curriculum further outperforms both two-stage variants by 7.4 and 11.8 points. By first learning to understand visual thoughts (Stage 1) and generate them reliably (Stage 2), Stage 3 can then focus on learning *when* visual thoughts are useful.

Table 6: Ablation of the three-stage training curriculum.

Method	Success Rate
Stage 3	0.273
+ Stage 1	0.345
+ Stage 2	0.301
+ Stage 1 + Stage 2 (APIVOT)	**0.419**

F.4 Visual Alignment Objective

Experiment Setup. To isolate the contribution of the visual-alignment loss $\mathcal{L}_{\text{vis}}$ applied in Stages 2 and 3 of training, we conduct an ablation in which we remove explicit alignment between the latent visual states and target features. Concretely, we keep the full three-stage curriculum and all cross-entropy objective terms fixed, and vary only the stages in which $\mathcal{L}_{\text{vis}}$ is applied.

Results. As shown in Table 7, removing $\mathcal{L}_{\text{vis}}$ entirely (None) decreases success by 15.5 points. In this variant, the model still learns to generate the visual-thought tokens due to the cross-entropy loss, but receives no supervision for its content. This performance drop shows that explicit visual alignment is important for learning meaningful visual thoughts for planning. Applying $\mathcal{L}_{\text{vis}}$ individually in Stages 2 and 3 improves over removing it entirely, but the full objective still performs best, indicating that $\mathcal{L}_{\text{vis}}$ contributes meaningfully in both training stages.

Table 7: Effect of the visual-alignment objective $\mathcal{L}_{\text{vis}}$ across training stages.

$\mathcal{L}_{\text{vis}}$ Supervision	Avg	CONTAIN	SORT	STORE
None	0.264	0.282	0.261	0.249
Stage 2 Only	0.373	0.429	0.364	0.326
Stage 3 Only	0.314	0.357	0.298	0.287
Stages 2 + 3 (APIVOT)	**0.419**	**0.472**	**0.421**	**0.365**

F.5 Modality-Label Placement and Supervision Frequency

Experiment Setup. APIVOT's third SFT stage teaches adaptive modality selection by training on reasoning traces where visual thoughts are inserted selectively for geometrically constrained subgoals. As described in Section 3.2 and Appendix A.2, these placements are determined based on simulator-derived heuristics. To isolate the effect of this placement strategy, we initialize all models from the same Stage 2 checkpoint and vary the dataset used for Stage 3 training. We compare our final heuristic-based dataset, constructed to use visual-thoughts 75% of the time, against a random baseline that inserts visual thoughts independently at each step with probability 0.75. This controls for the effectiveness of the geometric-heuristic. We also evaluate heuristic datasets with lower usage rates of 50% and 25%.

Results. As shown in Table 8, the heuristic-based 75% configuration achieves the best success rate. Although the Random baseline uses more visual thoughts at inference-time, it performs worse than our heuristic strategy. This suggests that the geometric heuristic provides an effective signal for learning to use visual thoughts adaptively when beneficial.

However, reducing the visual-thought rate to 50% and 25% degrades performance. Visual-thought usage appears to collapse at inference-time, with models using them on only 20% and 3% of steps. This suggests that sparse visual-thought supervision weakens the model's learned ability to use visual representations, even when initialized from a Stage 2 checkpoint that generated visual thoughts at every step. These results suggest the need for more robust strategies for learning adaptive modality selection.

Table 8: Reasoning trace heuristic ablation. **Bold** is the setting used to train APIVOT.

Reference Traces		Inference-Time	
Modality Strategy	Visual Thought Usage (%)	Success Rate	Visual Thought Usage (%)
Heuristic (Ours)	**75**	**0.419**	**56**
Heuristic	50	0.332	20
Heuristic	25	0.254	3
Random	75	0.371	63

F.6 Sensitivity to Geometric Heuristic Thresholds

Experiment Setup. Since the modality selection behavior is learned from the behavior demonstrated in the SFT traces, we analyze its sensitivity to the heuristics used to assign modality labels. In particular, each *place* action is assigned a visual thought if either one of two geometric heuristics, *free-space ratio* and *future-feasibility margin*, falls below a corresponding threshold. For APIVOT, we use thresholds of 0.65 and 3.8, respectively, chosen to maintain a sufficient frequency of visual-thought supervision for stable training while preserving meaningful variation in modality choice. In order to evaluate sensitivity to these thresholds, we vary each threshold independently by $\pm 50\%$ while keeping the other fixed.

Results. For each threshold configuration, we report the resulting fraction of training steps with a visual thought and inference-time success rate, number of tokens, and visual thought rate. As shown in Table 9, a 50% threshold change produces disproportionately large shifts in the fraction of visual thoughts in training, and consequently in model behavior.

Increasing either threshold by 50% causes nearly all steps to be visual, driving the model toward Always-Image behavior. By contrast, decreasing either threshold produces sparse visual supervision, with a resulting drop in both performance and inference-time visual-thought usage. This suggests that providing enough positive examples is needed to bootstrap modality selection. Under the default setting, APIVOT receives sufficient supervision to learn meaningful adaptive behavior and generalizes at test time.

Table 9: Sensitivity to geometric-heuristic thresholds. We vary each threshold independently by $\pm 50\%$ and report the training-data visual-thought (VT) rate and inference-time behavior.

	Training Data	Inference-Time Evaluation		
Setting	Visual Thought Rate	Success Rate	# Tokens	Visual Thought Rate
APIVOT (default)	**0.75**	**0.419**	**262**	**0.56**
Free Space $+50\%$	0.99	0.453	419	0.99
Free Space -50%	0.21	0.258	131	0.07
Feasibility $+50\%$	0.97	0.449	412	1.00
Feasibility -50%	0.37	0.295	168	0.13

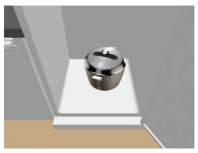
G Qualitative Examples

We provide qualitative comparisons against the top-performing VLM (Gemini-ER-1.5) and planning baseline (VLM-TAMP) across all three tasks: CONTAINMENT, SORTING, and STORING LEFTOVERS. Across these examples, APIVOT more reliably anticipates spatial constraints and produces physically feasible placements, leading to success while baselines struggle.

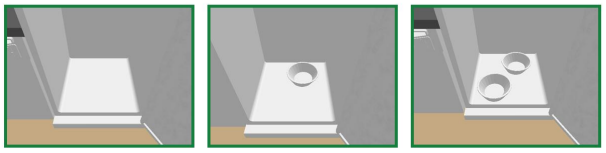
Task: Store the bowls inside the minifridge.

Explanation: APIVOT first clears the pot to create space for the bowls. Gemini-ER-1.5 recognizes the constraint too late, retries multiple placements, and times out.

Initial State:



APIVOT (Ours): Success

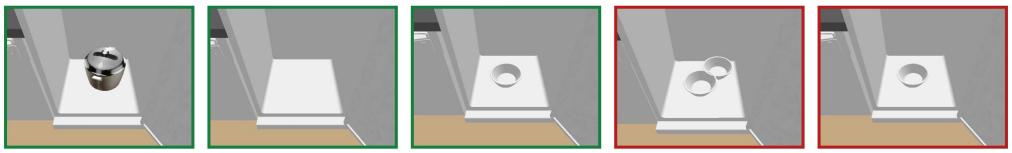


1. Clear pot | 2. Place bowl | 3. Place bowl

Gemini-ER-1.5: Failure



1. Place bowl (collides) | 2. Place bowl | 3. Place bowl (collides) | 4. Place bowl (collides) | 5. Place bowl (collides)



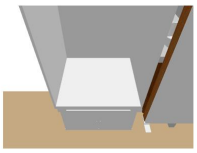
6. Remove bowl | 7. Clear pot | 8. Place bowl | 9. Place bowl (collides) | 10. Final state (timeout)

Figure 16: Qualitative comparison of CONTAINMENT: APIVOT clears an obstacle to complete the task, while Gemini-ER-1.5 recognizes the space constraint too late and times out.

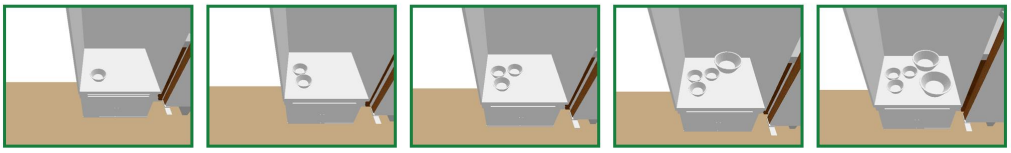
Task: Store the bowls inside the minifridge.

Explanation: APIVOT places bowls precisely while preserving space for future placements. VLM-TAMP fails to reserve enough room, making the final placement infeasible.

Initial State:

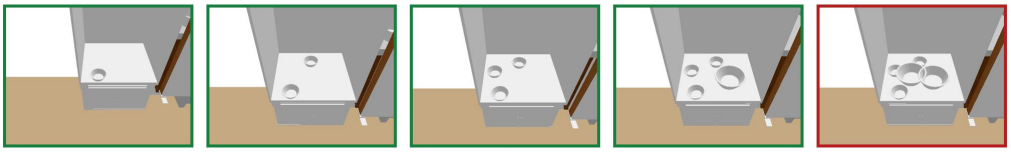


APIVOT (Ours): Success

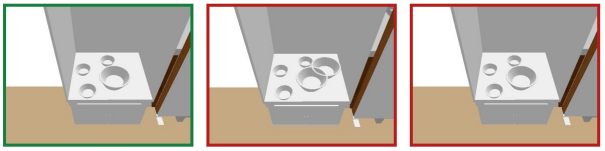


1. Place bowl | 2. Place bowl | 3. Place bowl | 4. Place bowl | 5. Place bowl

VLM-TAMP: Failure



1. Place bowl | 2. Place bowl | 3. Place bowl | 4. Place bowl | 5. Place bowl (TAMP failure)



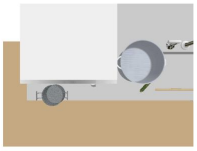
6. Move bowl aside | 7. Place bowl (TAMP failure) | 8. Final state (timeout)

Figure 17: Qualitative comparison of CONTAINMENT: APIVOT preserves space for future bowl placements, while VLM-TAMP fails to anticipate the final placement constraint.

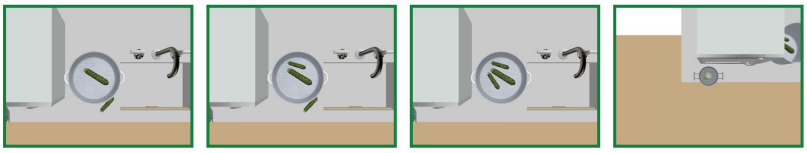
Task: Sort the zucchini and cabbage by type in the braisers.

Explanation: Both methods initially choose the correct braiser assignment. APIVOT succeeds through precise placement, while Gemini-ER-1.5 fails and then switches to an infeasible assignment.

Initial State:



APIVOT (Ours): Success

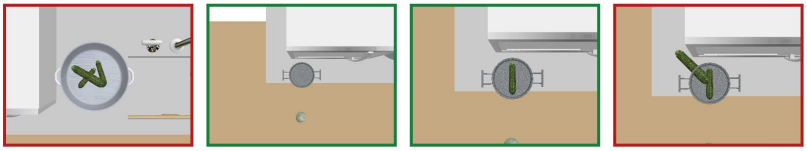


1. Place zucchini | 2. Place zucchini | 3. Place zucchini | 4. Place cabbage

Gemini-ER-1.5: Failure



1. Place cabbage | 2. Place zucchini | 3. Place zucchini | 4. Place zucchini (protrudes) | 5. Place zucchini (protrudes)



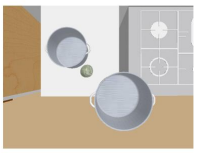
6. Place zucchini (collides) | 7. Remove cabbage | 8. Place zucchini | 9. Place zucchini (protrudes)

Figure 18: Qualitative comparison of SORTING: APIVOT succeeds with the correct braiser assignment, while Gemini-ER-1.5 fails and switches to an infeasible assignment.

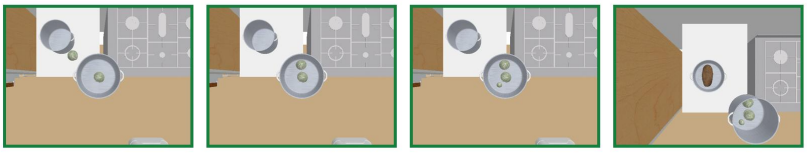
Task: Sort the cabbage and potato by type in the braisers.

Explanation: Although the potato is larger, there are more cabbages. APIVOT assigns cabbages to the larger braiser to allow more placement flexibility, while VLM-TAMP chooses the opposite assignment and fails.

Initial State:



APIVOT (Ours): Success

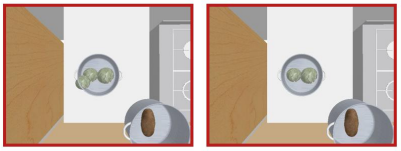


1. Place cabbage | 2. Place cabbage | 3. Place cabbage | 4. Place potato

VLM-TAMP: Failure



1. Place potato | 2. Place cabbage | 3. Place cabbage | 4. Place cabbage (TAMP failure) | 5. Shift cabbage



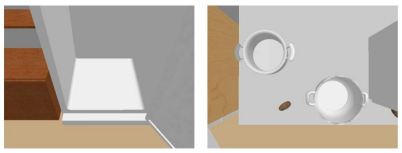
6. Place cabbage (TAMP failure) | 7. Final state (timeout)

Figure 19: Qualitative comparison of SORTING: APIVOT succeeds by selecting a more spatially flexible braiser assignment, while VLM-TAMP chooses the opposite, leaving insufficient room for placement.

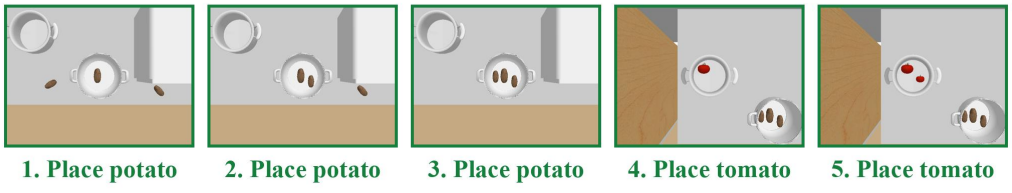
Task: Sort the potato and tomato by type in the braisers, and then store the braisers in the fridge.

Explanation: APIVOT sorts the objects and fits the two large braisers precisely in the fridge. Gemini-ER-1.5 sorts correctly but cannot fit both braisers inside.

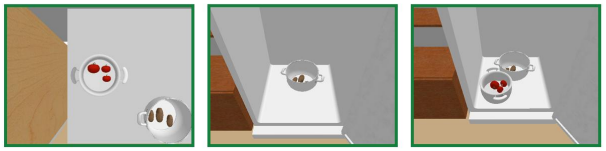
Initial State:



APIVOT (Ours): Success



1. Place potato | 2. Place potato | 3. Place potato | 4. Place tomato | 5. Place tomato

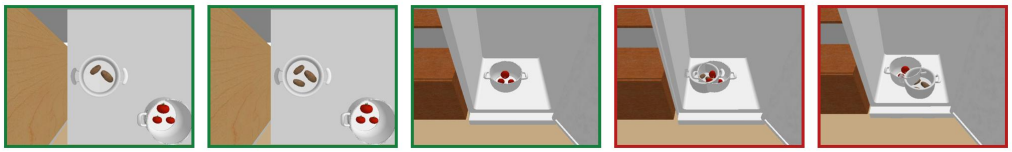


6. Place tomato | 7. Store braiser | 8. Store braiser

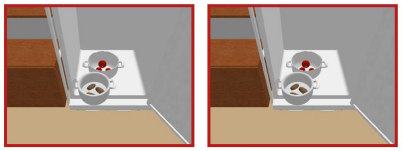
Gemini-ER-1.5: Failure



1. Place tomato | 2. Place tomato | 3. Place tomato (protrudes) | 4. Place tomato | 5. Place potato



6. Place potato | 7. Place potato | 8. Store braiser | 9. Store braiser (collides) | 10. Store braiser (collides)



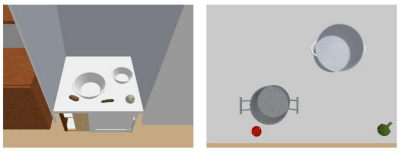
11. Store braiser (protrudes) | 12. Final State (model thinks its done)

Figure 20: Qualitative comparison of STORING LEFTOVERS: APIVOT sorts the objects and fits both large braisers in the fridge, while Gemini-ER-1.5 cannot place both braisers feasibly.

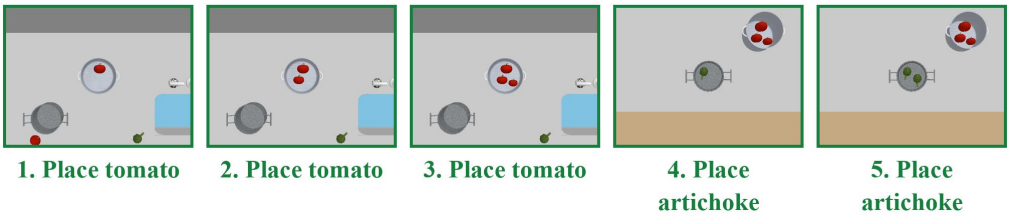
Task: Sort the tomato and artichoke by type in the braisers, and then store the braisers in the fridge.

Explanation: APIVOT sorts the food, removes obstacles, and places the braisers in a constrained fridge. VLM-TAMP's imprecise placements lead to repeated retries and eventual timeout.

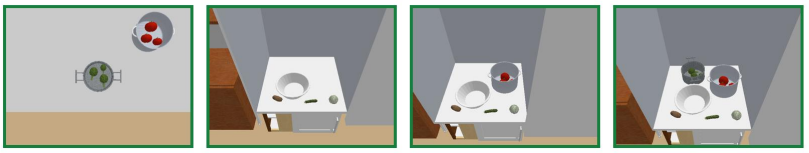
Initial State:



APIVOT (Ours): Success

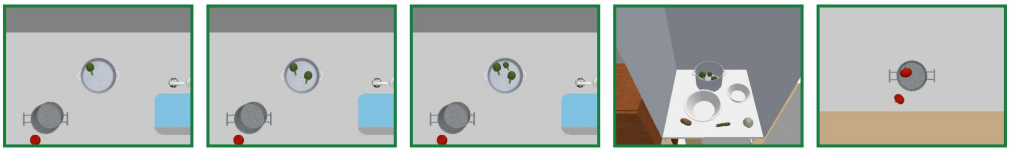


1. Place tomato | 2. Place tomato | 3. Place tomato | 4. Place artichoke | 5. Place artichoke

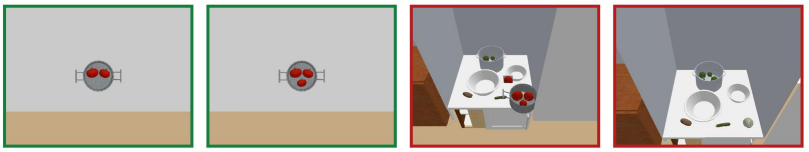


6. Place artichoke | 7. Remove blocking bowl | 8. Store braiser | 9. Store braiser

VLM-TAMP: Failure



1. Place artichoke | 2. Place artichoke | 3. Place artichoke | 4. Store braiser | 5. Place tomato



6. Place tomato | 7. Place tomato | 8. Place braiser (TAMP failure) | 9. Final state (timeout)

Figure 21: Qualitative comparison of STORING LEFTOVERS: APIVOT completes sorting, obstacle removal, and placement in a constrained region, while VLM-TAMP times out after several errors.

H Prompt Templates

Here, we provide the prompt templates for data generation and evaluation of VLM baselines.

H.1 Data-Generation LLM Expansion Prompt

We provide the prompt used to do trace expansion for the reference reasoning traces used for SFT. We use `gemini-3-flash-preview` for trace expansion.

Data-Generation Reasoning Prompt

```
You are given structured planning steps from a robot task-planning trace.  Your job is to convert each step into natural
reasoning that explains how a planner would think about the decision.

For EACH step, write 2-3 sentences of reasoning that follow this structure:
   1.  State what the action does in the context of the task.
   2.  Identify the key constraint that matters at this step.
   3.  Explain why the selected modality (text or text_image_post) is appropriate.

At the end, indicating that the task is complete and that it will output the final answer.  Do NOT add this earlier.
The reasoning should sound like a planner thinking through the decision, not like metadata description.

Reasoning style requirements:
   - Write in natural reasoning style.
   - Write reasoning in first-person planner voice (e.g.  'I need to...').
   - Do not simply restate the reason string.  Expand it into natural reasoning.
   - Keep reasoning aligned to each step.
   - Be concise but explanatory.


Output format:
Return the reasoning as JSON.
[{"step_index":  0, "reasoning":  "reasoning paragraph"},
   ...
   {"step_index":  n, "reasoning":  "reasoning paragraph + closing sentence"}]

Requirements:
   - Output ONLY valid JSON.
   - Do not include explanations outside JSON.
   - Do not include markdown.
   - Do not include commentary.
   - One entry per step.
   - Step indices must match input.
   - For the FINAL step only, append one additional closing sentence.


Example Input:
Step 0:
Action:  Move carrot into fridge
Action purpose:  place a goal object into the target container
Constraint type:  future_geometric_feasibility
Modality:  Text_image_post
Why this modality:  placement must preserve space for remaining objects

Example Output:
{"step_index":  0, "reasoning":  "I move the carrot into the fridge to place one of the goal objects into the target
container.  The key constraint is preserving enough room for the other vegetables.  A post-action image is useful
because I need to check whether the exact placement leaves space for future objects."}

----------------

Reminder of your goal:  For each step, you are expanding a structured planning decision into natural reasoning that
reflects how a task planner would think.  For the FINAL STEP ONLY, append an additional closing sentence.  The closing
sentence must clearly indicate that the task is complete and that it will output the final answer.  Do NOT add this to
any earlier steps.
Your reasoning should:
   - clearly connect the action to the task goal
   - explain the key constraint affecting the decision
   - justify why text or visual inspection is needed
   - sound like deliberate decision-making, not data description

Do not copy the template wording, list fields, repeat the reason string verbatim, invent new motivations, or add
unnecessary verbosity.  Focus on producing clear, natural, step-by-step planner reasoning that could realistically guide
execution of the task.

Now generate reasoning for the following steps.

{INSERT STEPS FOR A GIVEN TRACE}
```

Figure 22: Prompt used to expand structured planning traces into natural-language reasoning traces.

H.2 VLM Inference Prompt

We include the prompts used to evaluate planning for both our method (Figure 23) and text-only VLM baselines (Figure 24). In both prompts, the model is provided with the input observations, goal, object list, task constraints, and action primitives, and then prompted to output step-by-step reasoning, followed by a plan consisting of one formal action per line.

The prompt for planning with APIVOT specifies the output as a structured `<think>` trace, where each step specifies the planning intention, key constraint, modality decision, and justification, followed by a formal `<answer>` plan.

APIVOT Planning Prompt

```
You are given an input image of a kitchen world scene.  Your task is to reason about the image and plan
a sequence of actions that accomplishes the following goal:
‘‘{goal}’’.
Respond with actions in a formal language defined by the following primitive actions:
{set_of_actions}.
Each line must contain exactly ONE action.
Scene Objects:
Objects referenced in the actions must come from the following list.  Each name is formatted as
‘<object_category>_<instance_id>’.  Each object is paired with the position of its center in the format
[x, y] (normalized to 0-1000):
{object_id_to_points}
{task_constraints}
Planning Guidelines:
You are a mobile robot with {n_arms} arms.  You must obey the following commonsense rules:
1.  You must have at least one empty hand before you can pick up an object.
2.  You can only take actions on objects listed above.
3.  You should aim to achieve the goal with the minimum number of actions.
Answer Format:
You must output exactly two sections:
<think>...</think>
<answer>...</answer>
----------------
1) <think>section:
In the <think>section, you must produce a structured planning trace as an ordered sequence of reasoning
steps written in the format:
Step k:  ...
Each step should describe the next planning subgoal required to complete the task.  Each step should
include reasoning about the following:
1.  Planning intention:  State what must be decided or done next and how it contributes to achieving
the goal.
2.  Key constraint:
Identify the most important constraint affecting this step.  Examples include:
- spatial capacity
- object accessibility
- ordering dependencies
- none
3.  Modality decision:
Decide whether this step requires visual verification or whether text reasoning is sufficient.
4.  Justification:
Explain WHY that modality choice is appropriate.
----------------
Image representation rule:
If visual verification is required for a step, you must emit an image representation immediately after
the reasoning for that step.
Correct structure example:
Step 2:  I need to determine whether the container still has enough space for the remaining goal
objects.  The key constraint is preserving sufficient free space for future placements.  I need a
post-action image to verify that the remaining layout will leave enough free space for the other
objects to fit.  <IMAGE_START><IMAGE_PAD>...<IMAGE_END>
----------------
2) <answer>section:
Output the final plan as a list of primitive actions.
Requirements:
- One action per line
- Use only the formal primitive action language
- Do not include reasoning here
- The plan must be consistent with the reasoning in <think>
----------------
Final rules:
- Your output must follow EXACTLY this structure:
<think>...</think>
<answer>...</answer>
- Do NOT include anything outside these sections.
- Do NOT include explanations outside <think>.
- Do NOT include reasoning inside <answer>.
- Ensure the reasoning and plan are consistent.
```

Figure 23: Planning prompt used for APIVOT.

Text-Only VLM Evaluation Prompt

```
You are given an input image of a kitchen world scene.  Your task is to reason about the image and plan a sequence of
actions that accomplishes the following goal:
‘‘{goal}’’.
Respond with actions in a formal language defined by the following primitive actions:  {set_of_actions}.
Each line must contain exactly ONE action.
Scene Objects:
Objects referenced in the actions must come from the following list.  Each name is formatted as
‘<object_category>_<instance_id>’.  Each object is paired with the position of its center in the format [x, y]
(normalized to 0-1000):
{object_id_to_points}
{task_constraints}
Planning Guidelines:
You are a mobile robot with {n_arms} arms.  You must obey the following commonsense rules:
1.  You must have at least one empty hand before you can pick up an object.
2.  You can only take actions on objects listed above.
3.  You should aim to achieve the goal with the minimum number of actions.
Output Format:
First, write step-by-step reasoning about how to complete the task, including task decomposition, subgoal setting, and
action sequencing.  Each step should focus on planning the next planning subgoal required to complete the task, and
reasoning about what must be decided or done next and how it contributes to achieving the goal.
Then, write "Answer:" on its own line, followed by the final plan as a list of actions with one action per line.
```

Figure 24: Text-only planning prompt used for VLM baselines.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the abstract and introduction, we accurately summarize the scope and novelty of our work. We highlight the core idea of our method, provide an overview of the technical details, and discuss the key findings from our experiments.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our work in section 5.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: Our paper does not have theoretical results.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the details needed to reproduce experiments, including the data generation pipeline (Appendix A) and implementations of our model (Appendix B) and baselines (Appendix C). We will also be releasing our codebase with our model, baselines, dataset generation, and simulation pipelines.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

Justification: We provide our codebase at the following anonymous repository link: `https://anonymous.4open.science/r/adaptive_planning-F386/`.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Full details of our experimental settings are provided in Appendix B.2.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We provide the sizes of our evaluations datasets, but we do not have errors bars or statistical significance tests.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, these details are provided in Appendix B.2.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We reviewed the NeurIPS Code of Ethics and confirm that our research conforms with it in every respect.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work focuses on high-level reasoning capabilities of machine learning models. While it has the potential to enable other models with potential societal consequences, we do not feel there are any direct consequences that must be specifically highlighted here.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The data and models released as a part of our work do not pose any high risks.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Our work uses the KitchenWorlds simulator, which we cite in the paper. We also cite all models and baselines that we tested.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Yes, we will document our model and datasets in our codebase.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not run any crowdsourcing experiments or research with human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We do not have any studies with human participants.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We use gemini-3-flash-preview to expand structured planning skeletons into natural-language reasoning traces for supervised fine-tuning, as described in Appendix A.2.