SELECTIVE PLANNER ROUTING FOR EFFICIENT HIERARCHICAL ROBOT PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

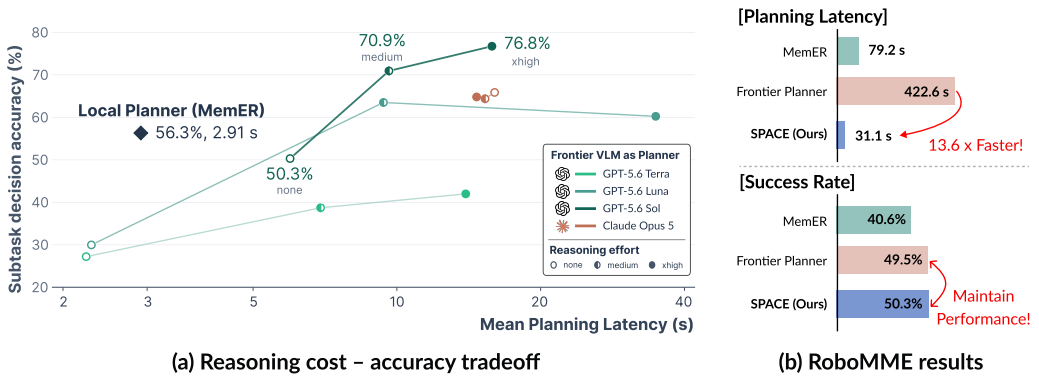
Hierarchical robot policies enable complex, long-horizon tasks by combining multimodal reasoning for high-level planning with low-level control. However, querying an expensive planner at every planning step incurs unnecessary latency, while relying solely on a lightweight planner can lead to incorrect subtask decisions when stronger reasoning is required. We introduce Subtask Planning with Adaptive Computational Effort (SPACE), a two-step framework that pairs a lightweight local planner with a compute-heavy frontier planner to decide when to update a subtask and when stronger reasoning is needed. The local planner first predicts a binary transition token; if it indicates retention, decoding stops and the robot continues executing the current subtask. Otherwise, the local planner predicts the next subtask, and a confidence-based router either accepts that prediction or invokes the frontier planner. To make that choice, the router considers token probabilities and how much plausible alternative predictions alter the proposed subtask's meaning or numerical details. Compared with a hierarchical baseline that predicts a full subtask at every planning step, SPACE raises task success from 40.6% to 50.3% on RoboMME and from 28.6% to 38.5% on RoboDojo, while reducing high-level planning time by 60.7% and 31.6%, respectively. On three challenging real-world manipulation tasks, SPACE also outperforms the hierarchical baseline in task success, highlighting its effectiveness across diverse reasoning demands.

1 INTRODUCTION

Long-horizon and complex robot tasks often require high-level planning in addition to low-level action generation, and hierarchical policies separate high-level planning from visuomotor control, allowing a planner to guide execution as the task progresses. In particular, Vision-language models (VLMs) are well suited to this role because they can ground planning decisions in visual observations and task instructions (Li et al., 2025; Shi et al., 2025; Team et al., 2025). Yet the repeated use of such planners makes inference latency as consequential as reasoning capability. Fine-tuned, lightweight VLMs, which we call *local planners*, therefore support frequent decisions at relatively low cost (Li et al., 2025; Sridhar et al., 2026). By contrast, more capable VLMs, which we call *frontier planners*, offer stronger reasoning for decisions involving task history and constraints but incur greater latency (Galanti et al., 2026; Hu et al., 2026).

However, choosing between the local and frontier planners matters only when the robot needs a new subtask; otherwise, it can continue executing the current one. Indeed, in our RoboMME (Dai et al., 2026) evaluation, the ground-truth subtask remains unchanged in 90.4% of scheduled planning calls (see section 4.3). When a new subtask is needed, selecting it may be straightforward or may require reasoning over task history and constraints. Consequently, predicting a full subtask on every call incurs unnecessary latency during retention, while querying the frontier planner for every transition adds latency to straightforward choices. Yet relying solely on the local planner risks errors on difficult ones. Figure 1 compares the local and frontier planners in subtask prediction accuracy and inference time. Although prior work uses model uncertainty to allocate additional computation (Cai et al., 2026; Li et al., 2026a) or determine when external assistance is needed (Ren et al., 2023; Wang et al., 2025), hierarchical planning requires addressing both decisions together: *When should the current subtask be retained, and when should choosing a new one involve the frontier planner?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Reasoning cost – accuracy tradeoff (b) RoboMME results

Figure 1: **Accuracy–latency trade-off across high-level planners and RoboMME results.** (a) Subtask prediction accuracy versus inference time for our MemER reproduction (Sridhar et al., 2026) and frontier VLMs. Higher planning accuracy generally requires longer inference time, highlighting the trade-off between local and frontier planners. (b) SPACE matches the empirical performance of frontier planner with significantly improved planning efficiency.

Our approach. To address both decisions, we introduce Subtask Planning with Adaptive Computational Effort (SPACE), a two-step framework that pairs a local planner with a frontier planner. First, the Local Planner predicts a binary transition token indicating whether to retain the current subtask. If the token indicates retention, decoding stops immediately and the low-level policy continues executing that subtask. Otherwise, the Local Planner predicts the next subtask. The Reliability Router then decides whether to accept this prediction or invoke the Frontier Planner. To make that choice, the Reliability Router considers token probabilities and how much plausible alternatives to individual tokens change the proposed subtask's meaning or numerical details. Thus, SPACE reserves full subtask generation for transitions and frontier reasoning for decisions on which the local prediction is less confident.

We evaluate SPACE on RoboMME (Dai et al., 2026), RoboDojo (Chen et al., 2026), and three real-world manipulation tasks. On RoboMME, compared with a hierarchical baseline that predicts a full subtask at every planning step, SPACE raises task success from 40.63% to 50.25% while reducing high-level planning time from 79.15 to 31.11 seconds per episode. The same comparison on RoboDojo shows success increasing from 28.58% to 38.48% and planning time decreasing from 63.80 to 43.62 seconds per episode. Beyond simulation, SPACE improves success over the baseline on three real-world tasks involving multistep rearrangement, arithmetic reasoning, and object tracking through occlusion. A paired timing replay further shows that stopping after the transition token reduces local planning time per episode by 81.8%, while routing analyses show that both token probabilities and changes in subtask meaning and numerical details contribute to planner selection.

Contributions. Our contributions are as follows:

- We characterize high-level planning effort through two linked decisions: whether to update the current subtask and whether predicting its replacement requires frontier reasoning.
- We introduce SPACE, which stops decoding when the current subtask can be retained and uses confidence-based routing between the Local Planner and the Frontier Planner when an update is needed.
- We demonstrate improvements in task success and planning time on RoboMME and RoboDojo, along with improved success on three real-world tasks. Timing and routing analyses examine the contribution of each decision.

2 Related Work

Hierarchical robot planning. Hierarchical robot policies connect high-level planning with low-level visuomotor control to perform complex tasks. Early work combines the task knowledge of

language models with learned robot-skill affordances to select executable skills from a predefined skill library (Ichter et al., 2023). Other approaches use language-level motions or semantic subtask predictions to mediate between task instructions and low-level actions (Belkhale et al., 2024; Black et al., 2025; Shi et al., 2025), or guide low-level policies through visual paths generated by high-level models (Li et al., 2025). Memory-based approaches further support context-dependent planning in long-horizon tasks by retaining and using execution history (Torne et al., 2026; Sridhar et al., 2026). Building on these hierarchical planning architectures, we study how to adaptively allocate high-level computation while maintaining subtask prediction accuracy.

Adaptive computation in hierarchical robot planning. Research on adaptive computation in robot planning adjusts when to reason and the amount or form of reasoning used for individual decisions. One line of work performs planning or reasoning only when execution progress indicates that a new decision is needed, and otherwise reuses previous results to reduce redundant computation (Ji et al., 2026; Lin et al., 2026; Li et al., 2026b). Other approaches apply additional search to high-level decisions when needed (Cai et al., 2026) or adaptively combine language and visual reasoning according to the semantic and geometric demands of the task (Jin et al., 2026). SPACE jointly considers when to generate a new subtask and which planner should handle it. Specifically, it uses the estimated reliability of the Local Planner's prediction to determine whether a more capable Frontier Planner is needed, thereby selectively allocating high-level computation.

Uncertainty-aware routing. Model routing and cascading allocate queries among models with different costs and capabilities using learned selection criteria or response reliability estimates (Ong et al., 2025; Chen et al., 2024; Chuang et al., 2025). In robot planning, calibrated uncertainty supports selective decisions between autonomous execution and human assistance, with conformal prediction and conformal-aware finetuning reducing unnecessary intervention (Ren et al., 2023). Related approaches improve reliability assessment by accounting for inherent task ambiguity (Liang et al., 2024) and distinguishing different sources of uncertainty (Yin et al., 2025). Prediction confidence also provides a criterion for allocating additional computation to uncertain high-level planning decisions (Cai et al., 2026).

Semantic uncertainty estimation. A complementary line of work incorporates semantic relationships among generated outputs into uncertainty estimation. These approaches account for shared meanings across different expressions (Kuhn et al., 2023) and finer-grained semantic similarities (Nikitin et al., 2024), providing uncertainty measures beyond surface-form variation. Our work connects reliability-aware model selection with semantic uncertainty estimation in hierarchical robot planning.

3 METHOD

We introduce SPACE, a hierarchical policy framework for efficient planning in long-horizon robotic tasks. We first introduce the overview of the proposed method dealing with different types of planners (Section 3.1). We then propose to reduce high-level planning latency through transition-aware subtask generation (Section 3.2) and confidence-based planner routing (Section 3.3). The overall framework is illustrated in Figure 2.

3.1 OVERVIEW

We consider long-horizon robotic tasks defined by global instruction I. Following prior work (Sridhar et al., 2026; Shi et al., 2025), we adopt a hierarchical policy comprising high-level planner (HLP) and low-level policy (LLP). HLP selects language subtasks that specify intermediate goals, and LLP executes them through visuomotor control.

At each high-level planning step t, HLP receives the instruction I, the previously selected subtask $\hat{g}_{t-1}$, and a visual planning context C_t. The context combines recent observations of the ongoing subtask with a visual memory comprising keyframes retained from earlier execution. The recent observations capture execution progress, while the stored keyframes preserve task-relevant information beyond the recent observation window. HLP selects the subtask to execute as $\hat{g}_t = \pi_{\text{HLP}}(I, C_t, \hat{g}_{t-1})$, where $\hat{g}_t$ denotes the *accepted subtask* passed to the LLP for execution. While executing $\hat{g}_t$, LLP produces a low-level action $a_k = \pi_{\text{LLP}}(o_k, \hat{g}_t)$, where k indexes low-

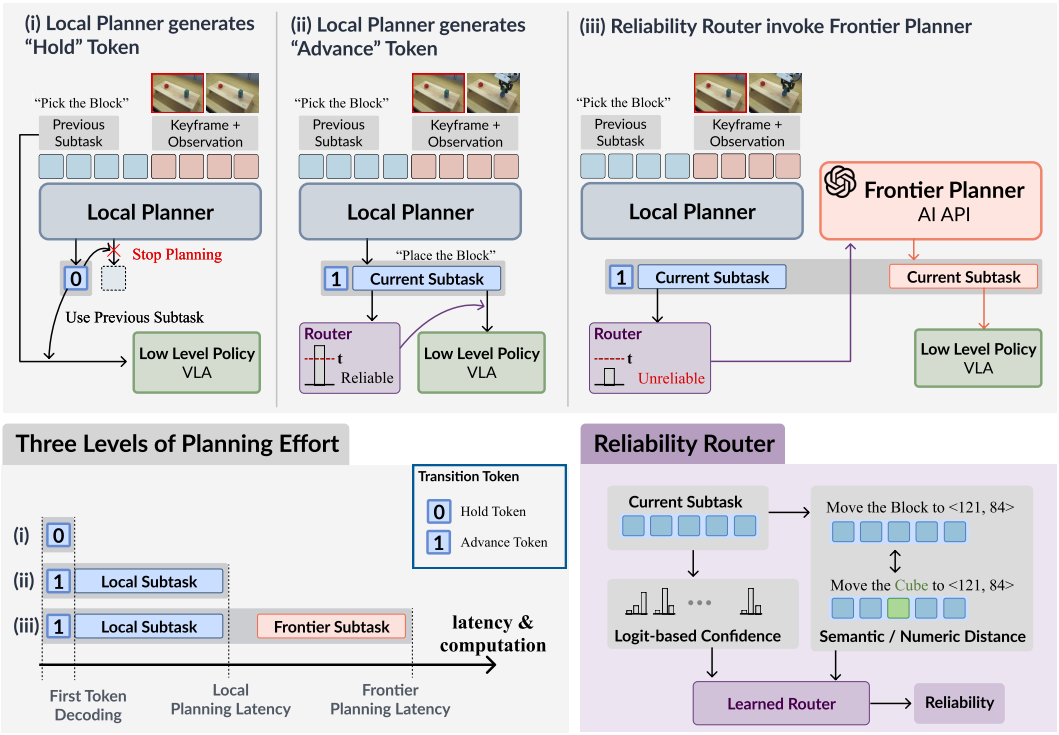


Figure 2: **Overview of SPACE.** Two sequential decisions yield three levels of planning effort. Local Planner first predicts whether a new subtask is needed. **(i)** If it outputs Hold, decoding terminates after the first token and the previously accepted subtask is reused. If it outputs Advance, decoding continues to generate a local subtask. Reliability Router then evaluates the correctness of the local subtask using token confidence and semantic and numeric differences from alternative subtasks. **(ii)** If the resulting reliability score exceeds the threshold, the local subtask is accepted. **(iii)** Otherwise, Frontier Planner generates a new subtask, which is accepted instead. In all three cases, LLP executes the accepted subtask.

level control steps. Our objective is to reduce high-level planning latency while maintaining subtask prediction accuracy.

SPACE uses a fine-tuned vision-language *Local Planner* as its default subtask planner and selectively invokes proprietary models as *Frontier Planner* through API calls. Local Planner first predicts whether to retain the current subtask or generate a new one. For advancing to new subtasks, a separately trained *Reliability Router* evaluates newly generated subtask to decide whether to accept it or request a prediction from Frontier Planner. This procedure avoids repeated subtask generation and reserves Frontier Planner calls for predictions judged unreliable.

3.2 Transition-Aware Local Planner

Local Planner is a Qwen3-VL-4B (Bai et al., 2025) fine-tuned to predict subtask transitions, generate subtask instructions, and select memory keyframes. It first predicts whether the currently executing subtask should be retained or a new high-level decision is required, avoiding redundant subtask generation while the current instruction remains appropriate. At each planning step t, it receives the global task instruction I, the visual planning context C_t, and the previously accepted subtask $\hat{g}_{t-1}$ being executed by the LLP.

Training. We derive transition labels from the dataset's ground-truth subtask annotations. We supervise the first output token with Advance at annotated subtask transitions and Hold otherwise. For Advance examples, the target sequence continues with the text instruction for next subtask, followed by the positions of recent frames to retain as memory keyframes. These positions index the recent observations from the preceding subtask interval. We fine-tune the model using the standard

causal language-modeling cross-entropy objective:

$$\mathcal{L}_{\text{local}} = -\sum_{j=1}^{N} \log p_\theta(u_j \mid x, u_{<j}),$$

where x denotes the planner input, $(u_1, \ldots, u_N)$ is the target output sequence, and θ denotes Local Planner parameters. This objective jointly supervises the transition token and, for $\mathrm{Advance}$ examples, the subsequent subtask text and frame positions.

Inference. At inference, Local Planner first emits $z_t \in \{\mathrm{Hold}, \mathrm{Advance}\}$. If $z_t = \mathrm{Hold}$, generation terminates immediately after this token, and LLP continues executing the cached subtask $\hat{g}_{t-1}$. If $z_t = \mathrm{Advance}$, the same decoding process continues to generate a new *local subtask* g_t^L, followed by the frame positions to store. The selected frames are added to visual memory for subsequent planning, following Sridhar et al. (2026). The generated subtask g_t^L is passed to the Reliability Router, which determines whether to accept it or invoke the Frontier Planner. Details of keyframe candidate construction and frame alignment are provided in Appendix A.1.

3.3 Candidate-Aware Reliability Router

Reliability Router is trained separately to predict whether a subtask g_t^L generated by Local Planner is correct. Its reliability score determines whether to accept the local prediction or invoke Frontier Planner for an additional prediction.

Correctness supervision. We construct training data by running the fine-tuned Local Planner on held-out episodes excluded from its fine-tuning. For each generated subtask, we collect the output token distributions and assign a correctness label $r_t \in \{0, 1\}$ by comparison with the ground-truth annotation. We set $r_t = 1$ when the action text matches the ground truth and, for subtasks containing coordinates, the absolute error along every coordinate axis is at most 10 pixels. Otherwise, we set $r_t = 0$. These labels are used to make Routers correctly predict the local subtask.

Candidate-aware features. Even though token confidence describes the model's distribution over candidate tokens, it does not directly quantify how selecting a different candidate would change the subtask. We therefore combine confidence features with semantic and numeric differences between the generated subtask and alternatives. These features are computed over the subtask text. Let $\mathbf{y}_t = (y_1, \ldots, y_L)$ denote the token sequence of g_t^L, and let $\mathcal{A}_i$ denote the candidate token set obtained from the next-token distribution at position i. The confidence features c_i are extracted from this distribution and include candidate probabilities. For each $v \in \mathcal{A}_i$, we construct an alternative subtask by replacing y_i with v while keeping all other tokens fixed:

$$\tilde{g}_{i,v} = (y_1, \ldots, y_{i-1}, v, y_{i+1}, \ldots, y_L). \tag{1}$$

This construction reuses candidates from the original generation pass and does not require generating each alternative.

To compare the original and alternative subtasks, we encode their semantic and numeric content separately. We mask numeric arguments and encode the remaining text with a pretrained Qwen3-Embedding model (Zhang et al., 2025). Numeric arguments are encoded using a deterministic representation adapted from Sundararaman et al. (2020). We combine these representations and compute the cosine distance $d_{i,v}$ between g_t^L and each $\tilde{g}_{i,v}$. The distances at position i form a feature vector d_i. Details of candidate selection and representation construction are provided in Appendix A.2.

Training. Router processes the sequence of concatenated features $([c_i; d_i])_{i=1}^{L}$ using a 3-stage module R_ω consisting of projection layer, a Transformer, and an MLP head. A final sigmoid σ maps the resulting output to a reliability score $q_t \in [0, 1]$:

$$q_t = \sigma\Big(R_\omega\Big(([c_i; d_i])_{i=1}^{L}\Big)\Big).$$

We train Router to estimate subtask correctness using binary cross-entropy while Local Planner's parameters being frozen:

$$\mathcal{L}_{\text{router}} = -r_t \log q_t - (1 - r_t) \log(1 - q_t).$$

Inference. Router is applied only when Local Planner predicts $\mathrm{Advance}$ and generates a new subtask. If $q_t > \tau_B$, the local subtask g_t^L is accepted. Otherwise, we invoke Frontier Planner to

Table 1: **Results on simulation benchmarks.** We report task success rate (%) and accumulated high-level planning time per episode (s) when using $\pi_{0.5}$ as the low-level policy on RoboMME and RoboDojo. Count., Perm., Refer., and Imit. denote the Counting, Permanence, Reference, and Imitation suites of RoboMME, respectively.

	RoboMME						RoboDojo	
Method	Time (s)	Count.	Perm.	Refer.	Imit.	Avg.	Time (s)	Avg.
$\pi_{0.5}$	–	27.00	22.00	15.00	10.00	18.50	–	21.10
w/ Local Planner (MemER)	79.15	**56.50**	46.50	40.50	19.00	40.63	63.80	28.58
w/ Frontier Planner	422.58	33.33	52.08	66.67	45.83	49.48	243.90	36.00
w/ SPACE (Ours)	**31.11**	39.50	**70.50**	**61.00**	**30.00**	**50.25**	**43.62**	**38.48**

generate the next subtask. The Frontier Planner receives the global task instruction I, the current image, and the full observation history sampled at 2 fps. A manually written system prompt provides task-specific knowledge, including coordinate conventions and task details omitted from the global instruction, to support zero-shot planning. The model returns only the next subtask, denoted by g_t^F, which is passed to the LLP for execution. We calibrate τ_B on the Router validation split to target a user-specified Frontier invocation rate B, with calibration details provided in Appendix A.2. The accepted subtask passed to LLP is therefore

$$\hat{g}_t = \begin{cases} \hat{g}_{t-1}, & \text{if } z_t = \text{Hold}, \\ g_t^L, & \text{if } z_t = \text{Advance and } q_t > \tau_B, \\ g_t^F, & \text{if } z_t = \text{Advance and } q_t \leq \tau_B. \end{cases}$$

Further details of Router architecture and threshold selection are provided in Appendix A.2.

4 Experiments

Our experiments address four questions:

- Does SPACE improve task performance while reducing high-level planning cost in long-horizon complex tasks? (Tables 1 and 2)
- How much computation can be avoided by determining whether a new planning decision is needed before full planning? (Figure 4)
- Is selectively routing difficult decisions more effective than uniformly increasing frontier-planner usage? (Figure 5)
- Which signals are necessary for reliable planner routing? (Table 3)

Implementation details. For all experiments, we use $\pi_{0.5}$ (Black et al., 2025) as a low-level policy. For high-level planning, We employ Qwen3-VL-4B (Bai et al., 2025) as a backbone for Local Planner, and we fine-tune it on structured planning examples derived from expert demonstrations. For Frontier Planner, we use GPT-5.6 Sol-XHigh (OpenAI, 2026). For features used for training Reliability Router, we compute semantic distances using a frozen 600M-parameter Qwen3 text-embedding model (Zhang et al., 2025), while coordinate differences are encoded separately. We provide additional implementation details in Sections A and B.

Baselines. We first compare against the single-stage policy $\pi_{0.5}$, which is conditioned on the global task instruction without hierarchical planning. For the local-planning baseline, we reproduce MemER (Sridhar et al., 2026) using the same backbone and amount of training data as our Local Planner, while retaining its original memory and planning formulation. We denote this baseline as "w/ Local Planner." We also evaluate a baseline that invokes the Frontier Planner at every high-level planning step, denoted as "w/ Frontier Planner."

4.1 Simulation Experiments

Setup. We first consider RoboMME (Dai et al., 2026), a benchmark for long-horizon, history-dependent manipulation comprising 16 tasks organized into four memory suites: Counting, Per-

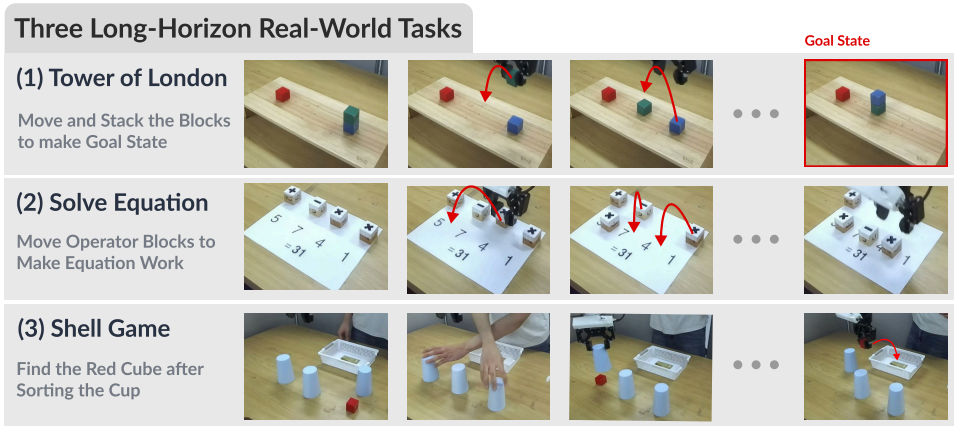


Figure 3: **Real-world evaluation setups.** Representative setups for Tower of London, Solve Equation, and Shell Game, involving block rearrangement, equation solving, and object tracking.

Table 2: **Real-world task success and planning time across three tasks.** Success rate (SR) and mean high-level planning time are reported separately for Normal and Hard settings. Planning time is the planner latency per episode of a successful rollout. MemER's time marked with * is estimated from its mean per-call latency on failed rollouts and the mean number of planner calls in training rollouts.

	Tower of London		Solve Equations		Shell Game	
Method	SR (%) ↑	Time (s) ↓	SR (%) ↑	Time (s) ↓	SR (%) ↑	Time (s) ↓
Normal Setting						
$\pi_{0.5}$	0.0	–	30.0	–	40.0	–
w/ Local Planner (MemER)	10.0	23.4	0.0	12.8*	20.0	30.4
w/ SPACE (Ours)	**70.0**	**10.7**	**90.0**	**11.6**	**73.3**	**30.3**
Hard Setting						
$\pi_{0.5}$	0.0	–	30.0	–	26.7	–
w/ Local Planner (MemER)	0.0	41.5*	0.0	**23.8***	26.7	36.3
w/ SPACE (Ours)	**40.0**	**18.7**	**40.0**	32.0	**46.7**	**35.2**

manence, Reference, and Imitation. We fine-tune the Local Planner on the training split of publicly released dataset, consisting of 100 expert demonstrations per task and 1,600 demonstrations in total. For Router development, we use the official validation split, comprising 50 episodes per task and 800 episodes in total, none of which are used for Local Planner fine-tuning. We conduct the final closed-loop evaluation on the official test split, comprising 50 episodes per task and 800 episodes per method. We also consider RoboDojo (Chen et al., 2026), a bimanual manipulation benchmark covering diverse visuomotor and reasoning tasks, and evaluate on all eight tasks in its long-horizon category. For controller training, we use 3,500 demonstration episodes collected in the RoboDojo simulator with a bimanual ARX-X5 robot. The hierarchical and single-stage policies use the same training episodes and frames, conditioned on frame-level subtask instructions and episode-level global instructions, respectively. We evaluate all methods on the same 25 deterministic layouts per task, yielding 200 evaluation episodes per method (Appendix C.1).

Results. In Table 1, SPACE achieves performance comparable to, or slightly better than, the Frontier Planner variant while substantially reducing high-level planning time. On RoboMME, SPACE achieves 50.25% task success compared with 49.48% for the Frontier Planner, while reducing planning time 13.6×. SPACE also achieves 9.62%p higher task success than MemER while requiring 2.5× less planning time. We also find consistent results on RoboDojo: SPACE achieves a slightly higher average success rate than Frontier Planner (38.48% vs. 36.00%) while requiring 5.6× less planning time. These results demonstrate that selective stronger-model assistance improves planning accuracy while retaining substantially lower execution cost than repeated full high-level planning.

Table 3: **Ablation study of the components of SPACE.** We examine (a) the effect of transition-aware subtask generation on success rate (SR; %) and accumulated high-level planning time (s), and (b) the effect of Reliability Router design choices on task success and Frontier Planner usage (%).

(a) Transition-aware local planning.

Method	Transition Token	SR ↑	Plan Time ↓
MemER	×	**40.63**	79.15
w/ Transition Token	✓	39.88	**13.03**

(b) Reliability router components.

Learned Router	Confidence	Semantic + Numeric	SR ↑	Frontier Usage ↓
×	✓	×	45.63	45.84
✓	✓	×	46.50	42.93
✓	×	✓	44.25	43.99
✓	✓	✓	**50.25**	43.69

4.2 Real World Experiments

Setup. Our real-world experiments use a DROID (Khazatsky et al., 2024) setup comprising a Franka Research 3 arm equipped with a Robotiq 2F-85 gripper. The LLP receives two 224×224 RGB image observations and a 10-dimensional robot state including cartesian position and 6d rotational representation (Zhou et al., 2019) of the end-effector, and gripper open/close. It operates in a 10-dimensional action space using delta end-effector control.

Tasks. To evaluate SPACE under varying high-level reasoning demands, we design three real-world manipulation tasks, illustrated in Figure 3:

- *Tower of London.* The robot rearranges and stacks multiple blocks to reach a target configuration. Early moves require reasoning about future moves and stacking constraints, while planning becomes simpler as the goal is approached.
- *Solve Equations.* The robot selects and places operator blocks to construct a valid equation. It is a single-shot planning problem whose difficulty increases with the number of operators to place.
- *Shell Game.* The robot tracks and retrieves a hidden cube as the cups are rearranged. Tracking the cube's location through the sequence of swaps requires reasoning over the observation history throughout the task.

Demonstrations and Subtask Annotation. We collect 165 demonstration episodes through teleoperation: 60 for Tower of London, 60 for Solve Equations, and 45 for Shell Game. Subtask labels are generated automatically from recorded robot states and gripper commands. We identify grasp and release events from the rising and falling edges of the gripper command, respectively. Each event is associated with the fixture whose Cartesian y-coordinate is closest to that of the end effector at the event. Release events define the boundaries between subtask segments.

Evaluation. We evaluate each task under Normal and Hard settings. For each setting, we conduct 10 evaluation episodes for Tower of London, 10 for Solve Equations, and 15 for Shell Game, yielding 70 real-world rollouts per method. Further evaluation details are described in Section C.2.

Results. In Table 2, we find that SPACE consistently improves task success on the settings with easier difficulty (*i.e.*, Normal), achieving 70.0%, 90.0%, and 73.3% success on Tower of London, Solve Equations, and Shell Game, respectively, and substantially outperforming both $\pi_{0.5}$ and the local-planner baseline. Compared with the local-planner baseline, SPACE also reduces high-level planning time on Tower of London and Shell Game. In particular, we find that the advantage of SPACE becomes more pronounced on the Hard settings. While the local planner completely fails on Tower of London and Solve Equations and achieves only 26.7% success on Shell Game, SPACE maintains success rates of 40.0%, 40.0%, and 46.7%, respectively. These results indicate that selectively using the stronger planner is particularly beneficial when the local planner alone is insufficient for challenging subtask decisions, and highlight the effectiveness of SPACE in allocating high-level planning effort according to the difficulty of each decision.

4.3 Analysis and Ablation Studies

Effect of transition-aware subtask generation. In Table 3(a) and Figure 4, we investigate the effect of transition-aware subtask generation on planning efficiency. As shown in Table 3(a), adding the

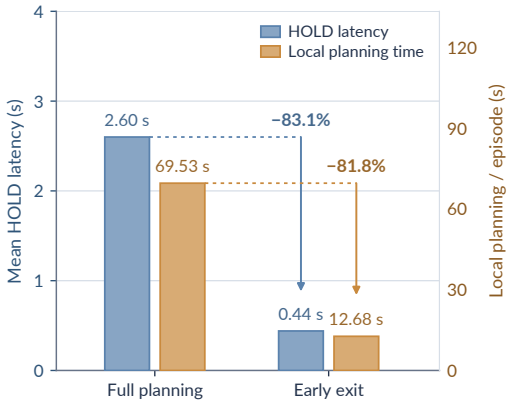


Figure 4: **Planning-time reduction from first-token termination.** We compare full decoding with first-token termination using the same checkpoint, differing only in whether decoding continues after the first token predicts HOLD.

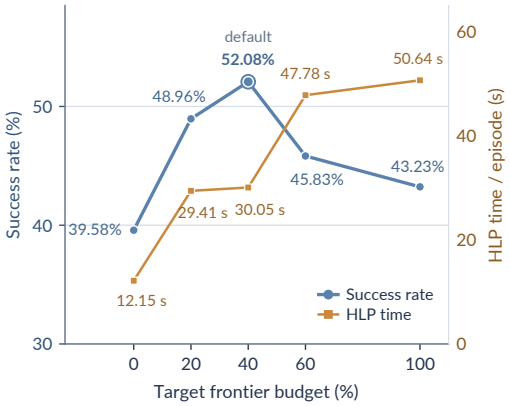


Figure 5: **Effect of the frontier-planner budget.** We vary the target fraction of new planning decisions routed to the Frontier Planner on a matched subset of RoboMME.

transition token reduces accumulated local planning time from 79.15s to 13.03s per episode while maintaining a similar task success rate (40.63 vs. 39.88%). We further compare Local Planner inference with and without terminating generation after the first output token for HOLD predictions. As shown in Figure 4, first-token termination reduces mean HOLD latency from 2.60s to 0.44s and accumulated local planning time from 69.53s to 12.68s per episode.

Router component analysis. In Table 3(b), we analyze which signals are useful for estimating the reliability of Local Planner predictions. A learned router using confidence features achieves 46.50% success, while semantic and numeric features alone achieve 44.25%. Combining confidence with semantic and numeric information achieves the highest success rate of 50.25% at similar Frontier Planner usage. This suggests that semantic and numeric differences provide complementary information to token-level confidence for planner routing.

Effect of frontier-planner budget. When a new subtask is needed, different planning decisions may require different levels of planning effort. In Figure 5, we investigate how the target Frontier Planner budget affects task success and high-level planning time. We vary the target budget while keeping the Local Planner, Frontier Planner, and Reliability Router fixed. Increasing the budget from 20% to 40% improves task success from 48.96% to 52.0%, whereas further increasing it to 60% and 100% reduces success to 45.8% and 43.2%, respectively, while high-level planning time increases. These results show that allocating more decisions to the Frontier Planner does not necessarily improve task success, highlighting the importance of selectively routing planning decisions based on Local Planner reliability.

5 CONCLUSION

We presented SPACE, a hierarchical planning framework that adaptively allocates high-level computation during long-horizon robot execution. SPACE separates frequent subtask-transition monitoring from full subtask generation and selectively invokes a stronger VLM only when the local planner is predicted to be unreliable. To support this decision, the router combines token-level confidence with semantic and numeric differences among alternative subtask predictions.

Across simulation benchmarks and real-world manipulation tasks, our results show that adaptive high-level computation can improve closed-loop task performance while reducing unnecessary planning cost. Further analysis shows that most high-level calls do not require generating a new subtask, that stronger reasoning is not uniformly beneficial across decisions, and that combining confidence with semantic information improves selective planner routing. These findings suggest that efficiently allocating reasoning across time and across planner capacity is an important direction for scalable hierarchical robot policies.

AI USE STATEMENT

In this work, we used generative AI tools to assist with drafting and polishing the manuscript and to generate portions of the research code. The research ideas, hypotheses, methodology, and experimental design were developed by the authors without generative AI assistance. We have reviewed all AI-assisted work. For example, all LLM-generated code was directly reviewed and verified for correctness by every author, and all AI-assisted text was reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Shuai Bai, Yuxuan Cai, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quan Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. RT-H: Action hierarchies using language. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.049.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Xiaowei Cai, Yunuo Cai, Bingao Chen, Jingxiao Chen, Zhi Chen, Siyuan Feng, Tengyu Hou, Jingshun Huang, Han Jiang, Runkun Ju, Dong Li, Mingxiang Li, Shaowei Li, Xinchen Li, Yifan Li, Yi Liu, Zhongyuan Liu, Jianlan Luo, Junwen Miao, Ruiqi Ni, Buqing Nie, Mingjie Pan, Xinlin Ren, Jianheng Song, Jiaxu Wang, Peiqi Wang, Sen Wang, Xiaoyan Wang, Dafeng Wei, Dongming Wu, Pengwei Xie, Pu Yang, Hangjian Ye, Xiangyu Yue, Jinyu Zhang, Qinglin Zhang, Xueyong Zhao, Pengfei Zhou, and Yue Zhou. τ_0-vla: A hierarchical robot foundation model with world-model-guided test-time computation. *arXiv preprint arXiv:2608.16885*, 2026. URL https://arxiv.org/abs/2608.16885.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. URL https://mlanthology.org/tmlr/2024/chen2024tmlr-frugalgpt/.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, Honghao Su, Zhiyang Dou, Kaixuan Wang, Dandan Zhang, Yunze Liu, Yan Qin, Qiwei Liang, Qiwei Wu, Zijian Lin, Wenwei Lin, Yuran Wang, Minghua He, Tianshu Wu, Ruihai Wu, Jingquan Zhou, Kai-Chong Lei, Haibao Yu, Yuanfeng Ji, Weiyang Jin, Guanyu Lin, Xiaofan Li, Qi Xiong, Renjing Xu, Zhongyu Li, Wenhao Chai, Enze Xie, Ziwei Wang, Yao Mu, Hao Dong, Wojciech Matusik, Mingyu Ding, Wenbo Ding, Ping Luo, and Masayoshi Tomizuka. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies, 2026. URL https://arxiv.org/abs/2607.04434.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route llms with confidence tokens. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10859–10878. PMLR, 2025. URL https://proceedings.mlr.press/v267/chuang25b.html.

Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. RoboMME: Benchmarking and understanding memory for robotic generalist policies. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/65933.

Liane Galanti, Dhruv Shah, and Tri Dao. Addressing the orchestration gap in generalist robots via physical agency. *arXiv preprint arXiv:2607.21725*, 2026.

Jiaheng Hu, Mohit Shridhar, Caden Lu, Dhruv Shah, Hao-Tien Lewis Chiang, Jie Tan, and Annie Xie. What matters in orchestrating robot policies: A systematic study of hierarchical vla agents, 2026. URL https://arxiv.org/abs/2606.10267.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T. Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Li Ji, Siyin Wang, Pengfang Qian, Xiaopeng Yu, Yihai Tian, Zhaoye Fei, Jingjing Gong, and Xipeng Qiu. HiMe: Hierarchical embodied memory for long-horizon vision-language-action control. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/60897.

Emily Jin, Joy Hsu, Yiqing Xu, Weiyu Liu, Nick Haber, and Jiajun Wu. Apivot: Adaptive planning with interleaved vision-language thoughts. *arXiv preprint arXiv:2607.08024*, 2026.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Wenhao Li, Xiu Su, Yichao Cao, Hongyan Xu, Xiaobo Xia, Shan You, Yi Chen, and Chang Xu. Vla-attc: Adaptive test-time compute for vla models with relative action critic model. In *International Conference on Machine Learning*, 2026a. URL https://icml.cc/virtual/2026/poster/61157.

Wenhao Li, Xiu Su, Dan Niu, Yichao Cao, Hongyan Xu, Zhe Qu, Lei Fan, Shan You, and Chang Xu. Sentinel-VLA: A metacognitive vla model with active status monitoring for dynamic reasoning and error recovery. In *International Conference on Machine Learning*, 2026b. URL https://icml.cc/virtual/2026/poster/61750.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. HAMSTER: Hierarchical action models for open-world robot manipulation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3bfee3bc6639c36e6e7b058db909f760-Abstract-Conference.html.

Kaiqu Liang, Zixu Zhang, and Jaime Fernández Fisac. Introspective planning: Aligning robots' uncertainty with inherent task ambiguity. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2299. URL https://papers.nips.cc/paper_files/paper/2024/hash/8451a20c5a7e0ee5671dda28f7daf7f3-Abstract-Conference.html.

Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. OneTwoVLA: A unified vision-language-action model with adaptive reasoning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tWMfhoP3as.

Alexander Nikitin, Jannik Kossen, Yarin Gal, and Pekka Marttinen. Kernel language entropy: Fine-grained uncertainty quantification for llms from semantic similarities. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0283. URL https://papers.nips.cc/paper_files/paper/2024/hash/10c456d2160517581a234dfde15a7505-Abstract-Conference.html.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5503a7c69d48a2f86fc00b3dc09de686-Abstract-Conference.html.

OpenAI. GPT-5.6 Sol Model. https://developers.openai.com/api/docs/models/gpt-5.6-sol, 2026. Accessed: 2026-09-26.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 661–682. PMLR, 2023. URL https://proceedings.mlr.press/v229/ren23a.html.

Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54919–54933. PMLR, 2025. URL https://proceedings.mlr.press/v267/shi25d.html.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Scaling up memory for robotic control via experience retrieval. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/9da515b1ad19d032a7398f00f5ff9b0c-Abstract-Conference.html.

Dhanasekar Sundararaman, Shijing Si, Vivek Subramanian, Guoyin Wang, Devamanyu Hazarika, and Lawrence Carin. Methods for numeracy-preserving word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 4742–4753. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.384.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, Steven Bohez, Konstantinos Bousmalis, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Oscar Chang, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, David D'Ambrosio, Sudeep Dasari, Todor Davchev, Coline Devin, Norman Di Palo, Tianli Ding, Adil Dostmohamed, Danny Driess, Yilun Du, Debidatta Dwibedi, Michael Elabd, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Leonard Hasenclever, Nicolas Heess, Brandon Hernaez, Alexander Herzog, R. Alex Hofer, Jan Humplik, Atil Iscen, Mithun George Jacob, Deepali Jain, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Chase Kew, Jerad Kirkland, Sean Kirmani, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Jacky Liang, Yixin Lin, Sharath Maddineni, Anirudha Majumdar, Assaf Hurwitz Michaely, Robert Moreno, Michael Neunert, Francesco Nori, Carolina Parada, Emilio Parisotto, Peter Pastor, Acorn Pooley, Kanishka Rao, Krista Reymann, Dorsa Sadigh, Stefano Saliceti, Pannag Sanketi, Pierre Sermanet, Dhruv Shah, Mohit Sharma, Kathryn Shea, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Jost Tobias Springenberg, Rachel Sterneck, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Vincent Vanhoucke, Jake Varley, Grace Vesom, Giulia Vezzani, Oriol Vinyals, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Yuxiang Yang, Rui Yao, Sergey Yaroshenko, Wenhao Yu, Wentao Yuan, Jingwei Zhang, Tingnan Zhang, Allan Zhou, and Yuxiang Zhou. Gemini robotics: Bringing ai into the physical world, 2025. URL https://arxiv.org/abs/2503.20020.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. MEM: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026. URL https://arxiv.org/abs/2603.03596.

Jun Wang, Guocheng He, and Yiannis Kantaros. Probabilistically correct language-based multi-robot planning using conformal prediction. *IEEE Robotics and Automation Letters*, 10(1):160–167, 2025. doi: 10.1109/LRA.2024.3504233. URL https://doi.org/10.1109/LRA.2024.3504233.

Shiyuan Yin, Chenjia Bai, Zihao Zhang, Junwei Jin, Xinxin Zhang, Chi Zhang, and Xuelong Li. Towards reliable llm-based robots planning via combined uncertainty estimation. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2600. URL https://papers.nips.cc/paper_files/paper/2025/hash/6fcbfb3721c1781728b10c6685cc2f6c-Abstract-Conference.html.

Yanzhao Zhang, Mingxin Li, et al. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. URL https://arxiv.org/abs/2506.05176.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5738–5746. IEEE, 2019.

A Implementation Details

A.1 Local Planner and Transition Gate

We implement the local planner by finetuning a pretrained Qwen3-VL-4B model (Bai et al., 2025). At planning step t, its input consists of the global instruction I, persistent visual memory $\mathcal{M}_t$, recent visual context $\mathcal{R}_t$, and the currently accepted subtask $\hat{g}_{t-1}$. The persistent memory contains keyframes selected from previous execution segments, whereas the recent context contains observations sampled every 24 control steps starting from the acceptance of the current subtask, along with the current observation. The frames are provided in temporal order, and the planner receives only the currently accepted subtask rather than a textual history of previous subtasks.

We use the existing vocabulary tokens `0` and `1`, rather than introducing additional special tokens, to represent HOLD and ADVANCE, respectively. For `0`, decoding terminates after the first token. For `1`, the planner additionally generates a local subtask prediction and keyframe positions in $\mathcal{R}_t$. The initial planning step and the first planning step after each annotated subtask boundary are labeled `1`; the remaining planning steps within the same subtask are labeled `0`. The initial inference step is also forced to `1` because no subtask has yet been accepted.

Following MemER (Sridhar et al., 2026), keyframe candidates are defined at annotated subtask boundaries. A boundary that does not coincide with a sampled observation is aligned to the temporally nearest observation. For an ADVANCE prediction, the nominated observations are appended to $\mathcal{M}_t$ in temporal order, and the recent context is restarted after the new subtask is accepted. A HOLD prediction leaves the persistent memory unchanged.

A.2 Reliability Router

Let $p_i(v)$ denote the local planner's next-token distribution at output position i. We retain the two most probable tokens and any additional token with probability greater than 0.05:

$$\mathcal{A}_i = \mathrm{Top2}(p_i) \cup \{v : p_i(v) > 0.05\}. \tag{2}$$

For each candidate, the router records its probability and logit offset from the most probable candidate. The probability mass outside $\mathcal{A}_i$ and the entropy of the full distribution are added as position-wise confidence features. Each retained candidate is substituted for the selected token at its output position to construct an alternative subtask prediction.

To compute differences between the selected and alternative predictions, numeric arguments are masked before the remaining text is encoded by a frozen Qwen3-Embedding-0.6B model (Zhang et al., 2025). A numeric value $a \in [0, A]$ is represented using

$$\phi_A(a) = \left[\cos\left(\frac{\pi a}{A}\right), \sin\left(\frac{\pi a}{A}\right)\right], \tag{3}$$

adapted from Sundararaman et al. (2020). Encodings of multiple numeric arguments are concatenated, normalized by the square root of their number, and zero-padded to a fixed length. Predictions without numeric arguments use a zero numeric vector. The semantic and numeric components are concatenated with equal weight ($w = 0.5$) and ℓ_2-normalized. We use cosine distance between the resulting representations as the difference between the selected prediction and each alternative.

The router aggregates the position-wise confidence and distance features using a one-layer Transformer encoder with a learned summary token, followed by an MLP with a sigmoid output. Candidate sets and output sequences are padded and masked during batching. The resulting score q_t estimates the correctness of the local subtask. We calibrate the threshold τ_B on validation data to target a frontier budget B among ADVANCE predictions. The main experiments use $B = 40\%$, and the calibrated threshold is fixed during test rollouts.

A.3 Frontier Planner

We use GPT-5.6 Sol-XHigh as the frontier planner. It is invoked when the local planner predicts ADVANCE and the reliability score satisfies $q_t \leq \tau_B$. The frontier planner receives the global instruction, available visual observations, task-specific knowledge, and the permitted subtask set.

Table 4: Training configuration for the RoboMME local planner.

Hyperparameter	Value
Initialization	Qwen3-VL-4B-Instruct
Trainable parameters	Language-backbone LoRA, 33.0M (0.74%)
Frozen parameters	Vision encoder and multimodal projector
LoRA rank / α / dropout	16 / 32 / 0.05
Optimizer	Fused AdamW
β_1 / β_2 / ϵ	0.9 / 0.95 / 10^{-8}
Learning rate	10^{-4}
Weight decay	0.1
LR schedule	Cosine, 0.05 warmup ratio
Global batch size	28
Training	1,300 optimizer steps
Scheduler horizon	2,286 steps
Maximum sequence length	3,200
Precision	BFloat16
Compute	2 NVIDIA A100 GPUs

For RoboMME, visual observations are sampled at 2 fps; the other evaluation settings use their corresponding domain-specific observation configurations.

The frontier planner does not receive the local subtask, the currently accepted subtask, or a textual history of previous subtasks. It independently infers execution progress from the provided visual context and generates a frontier subtask in the same instruction format used by the local planner. The resulting frontier subtask replaces the local subtask and is passed to the low-level policy for execution.

B Training Details

B.1 Training the Local Planner

For RoboMME, we initialize the local planner from Qwen3-VL-4B-Instruct. We train the adapter on the structured outputs described in Appendix A.1. The training set contains 31,979 examples constructed from 1,600 expert episodes across the 16 RoboMME tasks, with 5,551 ADVANCE examples and 26,428 HOLD examples. Only LoRA parameters in the language backbone are updated; the vision encoder and multimodal projector remain frozen. Table 4 summarizes the training configuration.

The global batch size is obtained using a per-device batch size of 7 on two GPUs with two gradient-accumulation steps. The 2,286-step scheduler horizon corresponds to two full passes over the training set; following the MemER checkpoint convention, training is stopped at step 1,300. The LoRA adapter is applied to the attention and MLP projection layers, with gradient clipping at 1.0 and gradient checkpointing enabled.

B.2 Training the Reliability Router

We freeze the finetuned local planner and generate subtask predictions on episodes that were not used during its finetuning. For each ADVANCE prediction, we extract the token-level confidence features and semantic–numeric distances described in Appendix A.2. We assign a binary correctness label by comparing the predicted subtask with the corresponding ground-truth subtask using the criterion defined in Section 3.3. These features and labels form the dataset used to train and validate the reliability router.

B.3 Training the Low-Level Policy

All RoboMME methods use the same $\pi_{0.5}$ low-level policy initialized from the public $\pi_{0.5}$ base checkpoint (Black et al., 2025). We finetune it on 476,857 action-supervised samples from 1,600

Table 5: Training configuration for the RoboMME $\pi_{0.5}$ low-level policy.

Hyperparameter	Value
Initialization	$\pi_{0.5}$ base checkpoint
Training samples	476,857
Action horizon	20
Optimizer	AdamW
β_1 / β_2 / ϵ	0.9 / 0.95 / 10^{-8}
Weight decay	10^{-10}
Gradient clip norm	1.0
Learning rate	5×10^{-5}
Warmup	10,000 steps
Global batch size	32
Training	80,000 optimizer steps
EMA decay	0.999
Precision	BFloat16
Compute	2 NVIDIA A100 GPUs

expert episodes covering all 16 RoboMME tasks. Each sample conditions action generation on the global task instruction and the annotated grounded subtask, with an action horizon of 20 steps. We use the final step-79,999 checkpoint for evaluation. Table 5 reports the training configuration.

The image encoder is frozen during finetuning, while the remaining policy parameters and the symbolic subtask-conditioning components are optimized. The learning rate is linearly increased to 5×10^{-5} during warmup and remains at that value for the rest of training.

C Evaluation Details

C.1 Simulation Benchmarks

RoboDojo Setup. RoboDojo is a bimanual tabletop manipulation benchmark covering diverse visuomotor and reasoning tasks. We evaluate all eight tasks in its long-horizon category. These tasks require a sequence of subtask predictions, making them suitable for evaluating high-level planning over extended execution. We therefore focus on this category to assess the contribution of subtask planning to task performance. We run 25 episodes per task, yielding 200 closed-loop episodes in total. All methods use the same finetuned $\pi_{0.5}$ low-level policy and are evaluated on the same episodes. The high-level planner is queried every 16 simulator steps, while only the planning strategy differs across MemER, Frontier Planner and SPACE.

C.2 Real-World Task Evaluation Details

Environment Setup. Our real-robot experiments use a Franka Research 3 arm equipped with a Robotiq 2F-85 gripper. A ZED 2i stereo camera captures the workspace from an exterior viewpoint, while a wrist-mounted ZED Mini stereo camera provides a close-up view of object interactions.

Baselines. All three share the same low-level interface: every high-level action expands into a fixed pick-and-place sequence at fixed slot positions, so the executor never has to read task semantics and manipulation variability stays near constant across difficulties. Differences in performance therefore reflect planning rather than control.

Real-World Evaluation Protocol. For each difficulty setting, we generate 10 test instances for Tower of London and Solve Equation and 15 test instances for Shell Game. Each instance is evaluated once without retries, and all methods are evaluated on the same set of test instances. An episode is counted as successful only when the final task goal is fully achieved, with no partial credit. Planning time is measured as the accumulated HLP inference time during an episode and is averaged over successful episodes only. We report N/A when a method has no successful episodes for a given setting.

Tower of London. Three blocks, one red, one green and one blue, occupy three fixed positions and may be stacked at most two high, giving 42 reachable states. A high-level action moves the topmost block of one position onto another; the executor realises it as a single pick-and-place. Because only three blocks are involved, a stack of three never lies on an optimal path, so the height cap removes no solvable instance: optimal distances between height-two states are identical with and without it. The graph diameter is five moves. We split instances by optimal solution length: *Normal* covers one and two moves, *Hard* three or more. The collection set contains 84 episodes drawn from all seven occupancy layouts, and the held-out set contains 30 with no start–goal pair shared with collection.

Solve Equation. Digits and a target value are printed on the sheet and never move. A staging row holds three operator blocks, $+$, $-$ and $\times$, at fixed slots; the planner assigns blocks to the gaps between digits, leaving the rest unused. Evaluation follows standard operator precedence. *Normal* uses two digits and one gap, giving three arrangements; *Hard* uses three digits and two gaps, giving six. Every instance is constructed so that exactly one arrangement satisfies the equation. Instances are generated with balanced answer arrangements and with no two consecutive sheets sharing an answer, so that a positional or operator prior yields no advantage. Collection and held-out sets are disjoint at the level of the digit multiset, not the individual instance: an n-digit set appearing in collection is excluded from evaluation entirely, since seeing it once exposes the value of every arrangement over those digits. The splits are fixed before any results are observed and recorded with the release.

Shell Game. A ball is revealed under one of three identical cups and the cup is lowered. The planner then observes a sequence of swaps, each exchanging two cups, and must name the cup holding the ball. No swap is repeated back to back which also makes it impossible for the ball to stay untouched through a sequence of length two or more. *Normal* uses one or two swaps, *Hard* three to five. The final observation is identical regardless of where the ball is, so this is the only one of the three tasks whose answer cannot be recovered from the current image; it is our probe for belief-state maintenance. The chance rate is $1/3$ at every sequence length, so accuracy is directly comparable across difficulties within this task.

Evaluation protocol. Scoring is binary in every task; we award no partial credit. A Solve Equation response counts as correct only when the full assignment of operators to gaps is right, and a Shell Game response only when the named cup is the one holding the ball. For Tower of London, many instances admit several optimal plans, sowe do not compare against a single reference trace: a plan counts as correct if it reaches the goal state in the optimal number of moves. Numeric proximity to the target is never credited, since a wrong equation is a failed task regardless of how close its value falls, and near-miss density is a property we control when constructing instances rather than a property of the response. We report accuracy per task and per difficulty and never pool across tasks, whose chance rates differ.

D Per-Task Evaluation Results

Tables 6 and 7 present the full evaluation results on RoboMME and RoboDojo, respectively, including task success rates and accumulated high-level planning time for each evaluated method.

Table 6: **RoboMME task success and planning time.** Main results use 800 episodes across 16 tasks, while Frontier-Only is evaluated on a matched 192-episode subset. Planning time denotes accumulated high-level inference time per episode.

Method	Planning Time (s)	Success Rate (%) ↑				
		Avg.	Count.	Perm.	Refer.	Imit.
$\pi_{0.5}$	–	18.50	27.00	22.00	15.00	10.00
$\pi_{0.5}$ + Local Planner (MemER)	79.15	40.63	**56.50**	46.50	40.50	19.00
$\pi_{0.5}$ + Frontier Planner	422.58	49.48	33.33	52.08	66.67	45.83
$\pi_{0.5}$ **+ SPACE (Ours)**	31.11	**50.25**	39.50	**70.50**	**61.00**	**30.00**

Table 7: **RoboDojo performance and planning time across eight tasks.** Scores are averaged over 25 episodes per task. Planning time denotes accumulated high-level inference time per episode.

Method	Planning Time (s) ↓	Success Rate (%) ↑								
		Avg.	Classify Object	TicTac Toe	Make Kong	Put Bottle	Organize	Pen Holder	Egg Holder	Stacking Toy
$\pi_{0.5}$	–	21.10	36.6	2.0	**32.0**	64.0	16.0	16.2	2.0	0.0
MemER	63.80	28.58	3.6	27.6	12.0	92.8	45.0	40.2	**7.4**	0.0
SPACE (Ours)	43.62	**38.48**	**37.0**	**43.6**	28.0	97.6	**54.0**	**44.6**	3.0	0.0
Frontier Planner	243.90	36.00	27.4	64.2	20.0	81.4	53.0	34.6	7.4	0.0

E PROMPT EXAMPLES

We provide the model-level prompts used by the local and frontier planners. The local planner uses the same system prompt across tasks. For the frontier planner, we instantiate the developer prompt with a BinFill example from RoboMME; step-specific user messages containing the visual observations are omitted.

Local Planner System Prompt

```
You are a robot program that predicts actions. The current input
    images from the front-view camera shows the most recent actions the
     robot has executed. The past keyframes are selected frames of
    particular importance from all the actions the robot has executed
    so far. Based on these, output the current subtask the robot should
     execute and nothing else. Some tasks may have a video input for
    initial setup, some may not.

The supplied current subtask is the last exact subtask accepted by the
     controller. Decide whether CURRENT has moved beyond that subtask.
    Your first generated token must be exactly 1 for transition/advance
     or exactly 0 for non-transition/hold. Then write a newline and a
    JSON object with:
- current_subtask: if the first token is 1, the exact action to
    execute now; if it is 0, the action after the supplied current
    subtask. Write unpredictable only when that later action cannot yet
     be determined.
- keyframe_positions: 1-indexed positions in the recent input image
    list that are nearest to keyframes not already in past memory.
```

Frontier Planner Developer Prompt: BinFill Example

```
You are the high-level planner for one RoboMME robot rollout.

Task: BinFill
Global instruction: put one red cube and three green cubes into the
    bin, then press the button to stop

Stable task knowledge:
Fill the bin with exactly the requested numbers of red, green, and
    blue cubes. Track how many cubes of each color have already been
    deposited from the chronological execution observations. A cube
    pickup is followed by placing that cube into the bin. Once every
    requested count is satisfied, press the button.

Visual timeline contract:
- Demonstration and execution observations are uniformly sampled raw
    frames at 2 FPS.
- They are chronological observations, not selected or annotated
    keyframes.
- Demonstration observations occur before execution observations.
- On every later decision, previously shown observations retain their
    order and newly observed execution frames are appended after them.
- A separate CURRENT image follows the prior-observation timeline and
    is the state where the new command will execute.

Allowed exact-subtask templates:
- pick up the first red cube at <coordinate>
- pick up the first green cube at <coordinate>
- pick up the first blue cube at <coordinate>
- pick up the second red cube at <coordinate>
- pick up the second green cube at <coordinate>
```

```
- pick up the second blue cube at <coordinate>
- pick up the third red cube at <coordinate>
- pick up the third green cube at <coordinate>
- pick up the third blue cube at <coordinate>
- pick up the fourth red cube at <coordinate>
- pick up the fourth green cube at <coordinate>
- pick up the fourth blue cube at <coordinate>
- put it into the bin at <coordinate>
- press the button at <coordinate>

Coordinate contract:
- Each <coordinate> is an integer [y, x] pixel in the CURRENT 256 by
    256 front-camera image.
- y increases from top to bottom and x increases from left to right.
- Select the command template first, then return one CURRENT
    coordinate per <coordinate>, in phrase order.
- Do not copy coordinates from earlier observations.

Output contract:
- Return only the strict JSON fields exact_subtask and coordinates.
- exact_subtask must be one allowed template with literal <coordinate>
     placeholders.
- coordinates must contain the matching CURRENT [y, x] pairs in phrase
     order.
- Do not output reasoning or any additional field.
```