Reflective Planning: Vision-Language Models for Multi-Stage Long-Horizon Robotic Manipulation

Yunhai Feng[1] **Jiaming Han**[2] **Zhuoran Yang**[3] **Xiangyu Yue**[2] **Sergey Levine**[4] **Jianlan Luo**[4†]

[1]Cornell University [2]CUHK [3]Yale University [4]UC Berkeley
https://reflect-vlm.github.io

Abstract: Solving complex long-horizon robotic manipulation problems requires sophisticated high-level planning capabilities, the ability to reason about the physical world, and reactively choose appropriate motor skills. Vision-language models (VLMs) pretrained on Internet data could in principle offer a framework for tackling such problems. However, in their current form, VLMs lack both the nuanced understanding of intricate physics required for robotic manipulation and the ability to reason over long horizons to address error compounding issues. In this paper, we introduce a novel test-time computation framework that enhances VLMs' physical reasoning capabilities for multi-stage manipulation tasks. At its core, our approach iteratively improves a pretrained VLM with a "reflection" mechanism - it uses a generative model to imagine future world states, leverages these predictions to guide action selection, and critically reflects on potential suboptimalities to refine its reasoning. Experimental results demonstrate that our method significantly outperforms several state-of-the-art commercial VLMs as well as other post-training approaches such as Monte Carlo Tree Search (MCTS).

Keywords: Vision-Language Models, Long-Horizon Manipulation

1 Introduction

Complex multi-stage manipulation tasks remain a fundamental challenge in robotics [1, 2, 3], especially when they require reasoning about sophisticated physical interactions across extended time horizons. These tasks often involve intricate action sequences where each step must account for physical constraints and potential consequences, making them particularly challenging for planning systems. Success hinges on understanding both the immediate effects of actions and their long-term implications, adapting plans based on execution outcomes, and generalizing to novel scenarios.

While classical planning approaches, such as task and motion planning (TAMP) [4, 5], can in principle address such problems, their reliance on predefined symbolic representations and explicit state estimation makes them difficult to apply in settings without known models that require visual perception [6, 7]. This limitation has motivated the search for more flexible approaches to robotic planning. Recent advances in vision-language models (VLMs) have shown remarkable capabilities in processing visual scenes and natural language instructions by leveraging internet-scale knowledge [8, 9, 10, 11, 12]. These models can effectively parse complex visual environments and comprehend high-level task descriptions expressed in natural language, making them promising candidates for robotic planning problems [13, 14, 15, 16, 17]. However, state-of-the-art VLMs still struggle with complex physical reasoning tasks, and this limitation becomes particularly pronounced when precise physics concepts and long-horizon planning are involved [18, 19].

In this paper, we study how to effectively leverage VLMs' Internet-scale knowledge while addressing their limitations in physical reasoning and long-horizon planning. We focus on a challenging class of robotic manipulation problems that involve sequentially manipulating interlocking objects to achieve

[†]Project advisor. Correspondence to: Yunhai Feng <yunhaif@cs.cornell.edu>, Xiangyu Yue <xyyue@ie.cuhk.edu.hk>, Jianlan Luo <jianlanluo@eecs.berkeley.edu>.

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

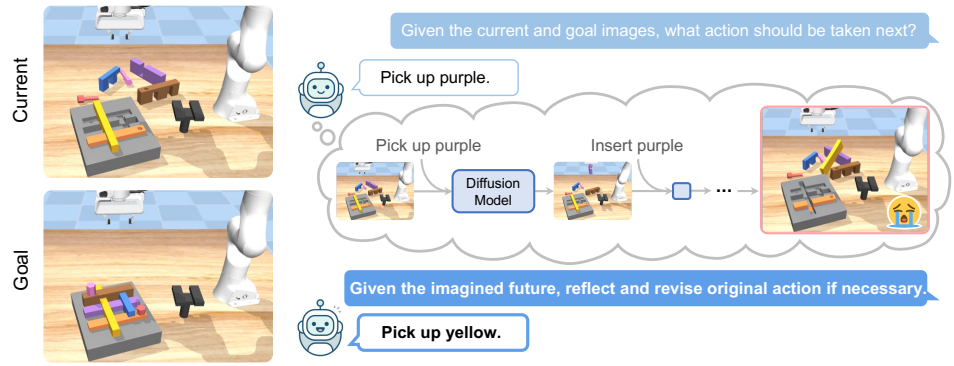

Figure 1. **Reflective planning.** Our method uses a VLM to propose actions and a diffusion dynamics model to imagine the future state of executing the plan. The imagined future helps the VLM reflect on the initial plan and propose better action.

desired configurations, as illustrated in Fig. 4. These tasks are particularly difficult as they require precise understanding of physical constraints, careful reasoning about action sequences, and the ability to plan over extended horizons while maintaining physical feasibility at each step.

To address these challenges, we present a novel test-time computation framework that significantly enhances VLMs' capabilities for multi-stage robotic manipulation tasks. The key insight of our method, ReflectVLM, is that by combining VLMs with a reflection mechanism and targeted post-training, we can create a system that better understands physical constraints and their implications for action planning. We use the term "reflection" to refer to a process where a VLM iteratively refines its decisions by critically examining the predicted outcomes of its proposed actions, akin to self-critique methods in large language models [20, 21, 22]. Our approach introduces two key components: (1) a look-ahead mechanism that uses a diffusion-based dynamics model to generate visual predictions of future states resulting from planned actions, and (2) a reflection process that allows the VLM to critique and refine its planned actions by analyzing these predicted outcomes. This combination of visual prediction and iterative refinement allows the VLM to develop a more sophisticated understanding of physical constraints and improve its decision-making capabilities without requiring extensive retraining.

Experimental results demonstrate that our approach significantly outperforms both the latest commercial state-of-the-art VLM models and traditional planning approaches like Monte Carlo Tree Search (MCTS) on this class of problems. Notably, our method achieves superior performance compared to post-training techniques such as supervised fine-tuning (SFT) while using the same amount of labeled data and maintaining computational efficiency. The success of our approach suggests that enhancing VLMs with structured reasoning mechanisms at test time can be a powerful strategy for improving their performance on physically-grounded tasks.

Our primary contribution is the mentioned test-time computation framework that enhances VLMs' physical reasoning capabilities for multi-stage manipulation tasks. Through extensive experiments, we demonstrate that our approach not only outperforms existing methods but also maintains computational efficiency. Importantly, while we demonstrate our framework's effectiveness on manipulation tasks, it is designed to be general and can be readily extended to other domains requiring visual understanding and sequential decision-making. This generality suggests broader applications in robotics and autonomous systems where physical reasoning and long-horizon planning are essential.

2 Related Work

Our framework incorporates a VLM with the reflection mechanism to solve long-horizon robotic planning problems. We therefore survey reflection techniques in the broader context in large models, VLM for robotic planning, as well as existing techniques for solving robot task and motion planning.

2.1 Reflection

Recent work has shown that large language models benefit from reflection mechanisms where models iteratively refine their outputs through self-critique and revision [23, 24, 25, 22, 26, 21, 20]. For

example, Madaan et al. [22] introduced an iterative refinement approach where models critique and improve their own outputs through self-feedback. Chain-of-thought prompting and its variants [27, 28, 29] demonstrated that guiding models to show their reasoning process leads to better performance. Such reflection mechanisms have also been extended to vision-language models [30, 31].

However, these approaches focus primarily on language-only or visual comprehension tasks, without addressing physical reasoning or robotics applications. Our work extends reflection to long-horizon robotic planning by incorporating a diffusion model that generates imagined future visual states. This allows the VLM to reflect on and revise its plans based on concrete visual predictions rather than relying solely on symbolic reasoning.

2.2 VLM for Robotic Planning

In robotics, several recent works have explored using VLMs for planning [13, 14, 15, 32, 33, 34, 35, 17, 16, 36]. However, these approaches either rely on symbolic state representations or make decisions in a single-step manner based only on current observations, without explicitly reasoning about future consequences or utilizing reflection mechanisms.

While ReplanVLM [37] and GameVLM [38] use VLMs to replan robot actions based on execution feedback, they still rely on symbolic state representations rather than visual imagination of future states. Black et al. [39] utilized a diffusion model to generate future visual states and executed them with a low-level goal-conditioned policy, but did not leverage these predictions for plan reflection or revision. Du et al. [40] combines a VLM with video prediction for beam search, but suffers from prediction error accumulation and struggles with physics-based reasoning tasks.

Our framework addresses these limitations by enabling VLMs to imagine and evaluate potential future states through a diffusion-based dynamics model. This allows for sophisticated multi-step planning while maintaining the benefits of VLMs' pre-trained visual-language understanding. The reflection mechanism further enables the VLM to critique and refine its plans based on these imagined futures, leading to more robust long-horizon manipulation.

2.3 Robotic Task and Motion Planning

Robotic Task and Motion Planning (TAMP) has been extensively studied [4, 5, 41]. Traditional approaches often combine symbolic planning with motion planning but struggle with real-world physical interactions and visual inputs. Learning-based methods [7, 6] show promise in handling uncertainty and complex dynamics but typically require significant task-specific engineering.

Our approach bridges this gap by leveraging VLMs' broad knowledge while adding structured physical reasoning through visual imagination and reflection. This enables robust long-horizon planning without requiring extensive task-specific engineering or large amounts of training data.

3 Preliminaries and Problem Statement

We formulate the multi-stage robotic manipulation planning problem as a partially observable Markov decision process (POMDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{Z})$. Here, $\mathcal{S}$ is the state space containing the full physical state of the environment, including object poses and physical properties; $\mathcal{A}$ is the action space consisting of high-level manipulation primitives $\{$`pick up`, `insert`, `reorient`, `put down`$\} \times \{\text{objects}\}$, assuming a failure rate ϵ for each primitive; $\mathcal{T}(s_{t+1}|s_t, a_t)$ represents the transition dynamics capturing physical interactions; $\mathcal{O}$ is the observation space of RGB images; and $\mathcal{Z}(o_t|s_t)$ is the observation model mapping states to image observations.

The objective is to find a policy π that generates actions to reach a goal state s_g. Due to partial observability, the policy only has access to image observations, taking the form $\pi(a_t|I_t, I_g)$, where I_t is the current observation and I_g is the goal image. The policy is instantiated as a VLM agent π_{VLM}, which takes a multi-modal input of images and text, and outputs textual action primitives.

Our framework includes pre-training and post-training phases, with the latter building on the framework of interactive imitation learning [42, 43] to learn a policy through environment interaction and real-time expert supervision. Thus under the standard assumption, we assume access to an interactive expert policy π_E that generates near-optimal actions $a^* = \pi_E(s)$ for any state s at training time.

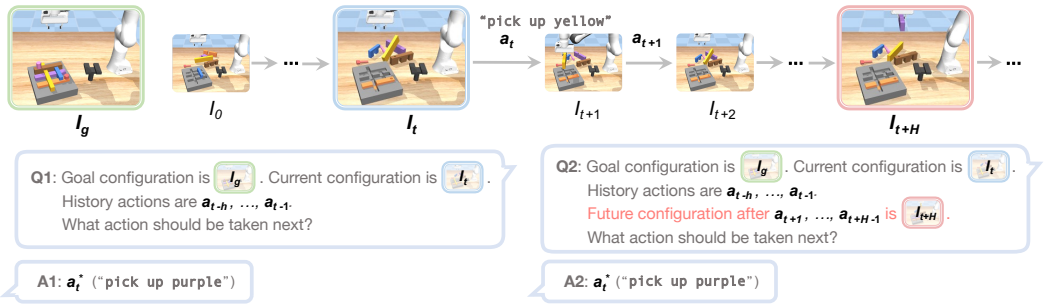

Figure 2. **Training data generation.** Training data for the reflection mechanism is collected by relabeling the rollouts. For each timestep, two training examples are generated: (Q1, A1) for action proposal and (Q2, A2) for reflection. a_t^* is the action label given by the expert policy. H is the imagination horizon, and h is the history length. We use $H = 5$ and $h = 10$ in experiments.

In this paper, we instantiated such an expert policy with access to the full state of the environment to generate optimal actions, though it could be obtained via other formats as well, e.g., human demonstrations. However, the VLM policy will only have access to image observations.

4 Reflective Planning with Vision Language Models

To address the challenges of physical interaction and long-horizon reasoning, we present a framework that incorporates VLMs with reflective planning. Our approach combines two key components: (1) a diffusion-based dynamics model that enables the VLM to imagine and evaluate future states, and (2) an interactive learning mechanism that allows the VLM to reflect on and revise its decisions based on these imagined outcomes. As shown in Fig. 1, these components work together to enable more robust manipulation planning while preserving the benefits of pre-trained VLMs.

4.1 Interactive VLM Policy Post-Training

While VLMs can generate actions based on visual inputs, they may hallucinate physically implausible solutions without actual interaction experience. To overcome this and enable long-horizon reasoning, we introduce an interactive learning algorithm that teaches the VLM to reflect on and improve its decisions through direct interaction with the physical environment. This process further enhances a base VLM policy, which is initially trained on a fixed set of expert demonstrations. Similar to DAgger [42], we iteratively collect new data by rolling out the VLM policy in the environment and finetune the VLM policy with the aggregated data. As formulated in Algorithm 1, N trajectories are collected in each iteration. At each timestep, we generate a learner action $a_t^\dagger$ by prompting the VLM with the images of the goal and current states, as well as an expert action a_t^* from the expert policy. The pairs $((I_g, I_t), a_t^*)$ are then added to the dataset for finetuning. To facilitate convergence, we execute the learner action $a_t^\dagger$ with a probability of p and the expert action a_t^* with a probability of $1 - p$, instead of always following the actions from the learner.

Algorithm 1 Interactive VLM Post-Training

Require: initial state distribution ρ_0, goal state distribution ρ_g, numbef of iterations K, number of trajectories per iteration N, episode length T, imagination horizon H, expert policy π_E, expert demonstrations $\mathcal{D}^*$

train base policy π_{VLM} on $\mathcal{D}^*$
$\mathcal{D} \leftarrow \mathcal{D}^*$
for $i \leftarrow 1$ to K **do**
  $\mathcal{D}_i \leftarrow \emptyset$
  // rollout out policy π_{VLM} to collect data $\mathcal{D}_i$
  for $n \leftarrow 1$ to N **do**
    $s_0 \sim \rho_0$; $I_0 \leftarrow \mathcal{Z}(s_0)$
    $s_g \sim \rho_g$; $I_g \leftarrow \mathcal{Z}(s_g)$
    for $t \leftarrow 0$ to $T - 1$ **do**
      $a_t^\dagger \sim \pi_{\text{VLM}}(I_g, I_t)$; $a_t^* \sim \pi_E(s_g, s_t)$
      $a_t \leftarrow a_t^\dagger$ **if** `random()` $< p$ **else** a_t^*
      $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$; $I_{t+1} \leftarrow \mathcal{Z}(s_{t+1})$
    end for
    $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{((I_g, I_t), a_t^*)\}$
    $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{((I_g, I_t, I_{t+H}, a_{t:t+H-1}), a_t^*)\}$
  end for
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
  finetune π_{VLM} on $\mathcal{D}$
end for

To generate training data for reflection, we can simply relabel a trajectory after it is terminated, as also illustrated in Fig. 2. Specifically, the image I_{t+H}, which is a future observation following the

action sequence $a_{t:t+H-1}$, is added to the context for reflection at timestep t, and the VLM is still supervised to output the same expert action a_t^*. Intuitively, this image provides additional information about the effect of executing the action sequence as a feedback, which can be leveraged by the VLM to decide whether the initially proposed action sequence leads to a promising future state.

In essence, we are generating two forms of question answering examples from interaction with the environment. The first is to predict an optimal action given images of the goal and current state, and the second is to reflect and revise an initial action sequence proposal by looking into an additional future image. Since a VLM can flexibly take any text and images as input, these two tasks can be handled by a single VLM with two different prompt templates, as summarized in Fig. 2. See App. I for full prompts, and App. H.1 for detailed VLM architecture. The VLM is trained to generate actions aligned with expert actions in the dataset with a cross entropy loss:

$$\min_{\pi_{\text{VLM}}} \mathbb{E}_{\mathcal{D}} \left[\mathcal{L}_{\text{CE}}(\pi_{\text{VLM}}^{\text{propose}}(a_t|I_g, I_t), a_t^*) + \mathcal{L}_{\text{CE}}(\pi_{\text{VLM}}^{\text{reflect}}(a_t|I_g, I_t, I_{t+H}, a_{t:t+H-1}), a_t^*) \right]. \quad (1)$$

4.2 Diffusion Dynamics Model

A key component in reflective planning is predicting future states accurately when evaluating potential action sequences. While our interactive learning mechanism enables the VLM to learn from physical interactions, we need an additional capability during inference—the ability to imagine and evaluate hypothetical futures without actually executing actions in the environment. To address this, we develop a diffusion-based dynamics model (DDM) that efficiently generates predicted visual observations by conditioning on the current observation and a proposed action sequence. This allows the VLM to simulate the consequences of its actions before committing to them.

Building on advances in diffusion-based generative models [44, 45, 46], we formulate the forward dynamics prediction as an image-to-image translation task. Our diffusion dynamics model takes the current observation I_t and action a_t as input to predict the next observation I_{t+1}. Rather than training a diffusion model from scratch, which would require substantial computational resources and training data, we leverage the pretrained Instructpix2pix model [47] that has been trained on large-scale image editing datasets as our base model.

Data. We curate a dataset for training the diffusion model. To encourage broader coverage of visisted states, the data collection policy is a noised version of the expert policy. Due to the difficulty of this task, we also include a few test data points to improve the fidelity and accuracy of the DDM. Details can be found in App. H.2.

Architecture. The model architecture is shown in Fig. 3. For the input (I_t, a_t), we first encode them into latent representation z_t and z_{a_t} with pretrained latent encoder and text encoder. Then we feed z_t, a sampled noise $\mathcal{N}$ and the action condition z_{a_t} into the diffusion UNet for de-noising. Finally, we decode the predicted z_{t+1} into a future observation I_{t+1} with a latent decoder.

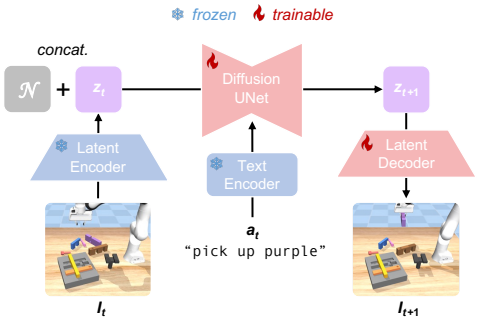

Figure 3. **Architecture of Diffusion Dynamics Model.** The latent encoder and text encoder are frozen during training, while Diffusion UNet and latent decoder are finetuned on our task data. $\mathcal{N}$: random noise.

Training. The training of DDM consists of two separate phases: UNet training and decoder training. The UNet training phase is to learn transformations from z_t to z_{t+1} conditioned on z_{a_t}, while the latent decoder training is to adapt the pretrained VAE models into our task domain because our task requires precise reconstruction of small pieces on the table. Since we keep the latent encoder frozen, we can train the two phases in parallel.

4.3 Reflective Planning

With the VLM policy trained via interactive learning and the diffusion model serving as a dynamics proxy to imagine future outcomes, we now introduce our reflective planning mechanism for decision making at inference time. Alg. 2 shows the detailed process. We use $\tilde{I}$ and $\tilde{a}$ to denote the generated

image and action, which are not actually observed or executed in the environment. To get the future image after H steps, where H is the planning horizon, we perform H iterations of action proposal and diffusion generation. At each iteration, the VLM policy is prompted by the goal image I_g and the generated image $\tilde{I}_{t+k}$ at the previous iteration to propose an action $\tilde{a}_{t+k}$. The diffusion model $\tilde{\mathcal{T}}$ then generates the future image $\tilde{I}_{t+k+1}$ conditioned on the previous image $\tilde{I}_{t+k}$ and the action $\tilde{a}_{t+k}$. For the first iteration, the input image $\tilde{I}_t$ is just the current observation I_t. After this process of imagination, the generated future image $\tilde{I}_{t+H}$ and the plan $\tilde{a}_{t:t+H-1}$ are concatenated with the goal and current observation, and fed into the VLM policy for reflection. The VLM policy will then output the final action a_t to be executed. Action proposal and reflection are performed by the same VLM policy with two different prompt templates, as indicated by the superscripts "propose" and "reflect".

Algorithm 2 Reflective Planning (Inference)

Require: current image I_t, goal image I_g, imagination horizon H

$\tilde{I}_t \leftarrow I_t$
for $k \leftarrow 0$ to $H-1$ **do**
  $\tilde{a}_{t+k} \leftarrow \pi_{\text{VLM}}^{\text{propose}}(I_g, \tilde{I}_{t+k})$
  $\tilde{I}_{t+k+1} \leftarrow \tilde{\mathcal{T}}(\tilde{I}_{t+k}, \tilde{a}_{t+k})$
end for
$a_t \leftarrow \pi_{\text{VLM}}^{\text{reflect}}(I_g, I_t, \tilde{I}_{t+H}, \tilde{a}_{t:t+H-1})$
Output: a_t

5 Multi-Stage Robotic Manipulation Planning Tasks

Inspired by Luo et al. [48], we procedurally generated a suite of multi-stage long-horizon manipulation tasks that require understanding of physical interactions and reasoning about the effects of long-term action sequences. The task is initialized with a board and a set of small pieces randomly placed on a table. The goal is to fully assemble the board by inserting the pieces into the board one by one. Examples of the initial and goal configurations are shown in Fig. 4. Detailed task generation process is included in App. C. Notably, most tasks include inter-locking pieces so that they can be inserted into the board only in a specific order. As an example, Fig. 4(b) shows the dependencies between the pieces in one of the tasks. This design tests an agent's ability to strategically select manipulation targets and recover from failures through replanning.

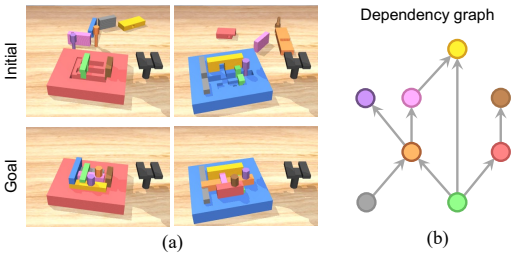

Figure 4. **Task examples.** (a) Two generated multi-stage manipulation tasks with interlocking pieces. Top: initial configurations. Bottom: goal configurations. See App. D for more examples. (b) The graph shows the dependencies between the objects in the blue assembly board on the left. Each node represents an object, and each directed edge indicates the predecessor object should be assembled before the successor object.

We focus on the high-level planning of this long-horizon manipulation task. We define a set of actions in the form of "`[act] [obj]`", where `[act]` $\in$ {`pick up`, `insert`, `reorient`, `put down`} is an action primitive, and `[obj]` denotes the object to be manipulated. See App. F for details of the action primitives. Each action primitive is implemented as a rule-based script controller; however, integrating other low-level controllers, such as learning-based policies like behavior cloning, is also possible. We also designed an expert policy with the mentioned motor primitives; see App. G for implementation details.

6 Experiments

Our experiments evaluate the effectiveness of our method and analyze its key components. We aim to answer three key research questions. First, how well does our method perform in long-term planning, particularly when handling complex physical interactions? Second, how effectively does our method generalize across different object configurations while maintaining the ability to reason and plan reactively in dynamic environments? Third, what is the impact of the reflection mechanism on the overall performance of our method? To address these questions, we conduct comprehensive experiments comparing ReflectVLM against: (1) state-of-the-art VLM models tested in zero-shot fashions, (2) model-based planning approaches like MCTS, and (3) ablation studies examining the reflection mechanism.

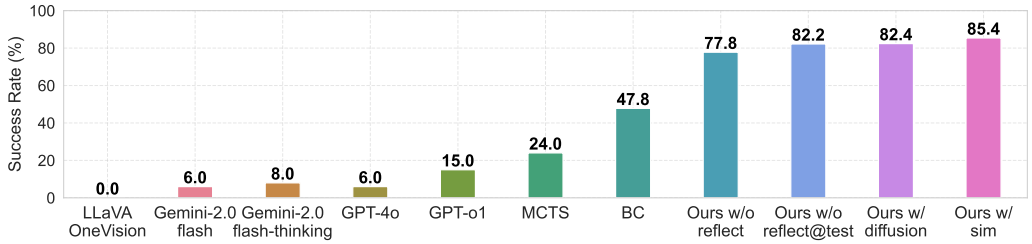

Figure 5. **Performance of our method and baselines.** Success rate (%) on 100 tasks. For the zero-shot test of state-of-the-art VLMs and MCTS, the experiments were conducted once; for other methods, the results are the average of five seeds.

6.1 Experiment Setup and Policy Training

To evaluate the generalization capabilities of different models, we generate two distinct task sets: a training set using the procedure described in Sec. 5, and a separate evaluation set containing previously unseen configurations. The evaluation tasks are specifically designed to test generalization across varying object configurations, colors, and spatial arrangements. We particularly emphasize challenging scenarios that require sophisticated physical reasoning and multi-step planning. For instance, some tasks begin with objects in physically obstructing positions that prevent direct task completion - requiring the policy to first remove the obstructing pieces and then develop a new plan for the original objective. Specifically, the training set contains 1000 different tasks, each generated task was randomized to five different initial spatial arrangements, these tasks are used to pre-train the VLM policy. At each iteration of post-training, we randomly sample 200 out of these 1000 tasks to further train the VLM policy with the reflection mechanism. The evaluation set contains 100 different tasks that are unseen in the training set; see App. E for statistics of the evaluation tasks.

As mentioned in Sec. 3, our method utilizes an oracle policy operating in the environment's symbolic state space to generate expert demonstrations for training. This oracle achieves a 97% success rate across tasks, but importantly, it operates with access to ground-truth state information. In contrast, our VLM policy must rely solely on visual observations. While alternative data sources like human demonstrations could be used for training, we chose this oracle-based approach to systematically study our method's capabilities under controlled conditions.

During the policy pre-training phase, we utilize the oracle policy to provide action labels, then finetune an LLaVa-1.5-13B model [12, 49] with standard supervised learning loss. This pre-training used 5,000 expert demonstrations (1,000 unique tasks × 5 initial configurations per task). In the post-training phase, we use the same oracle policy to further train the VLM policy from the previous stage using the procedure described in Alg. 1. For each iteration of post-training, we collect 1k trajectories by rolling out the VLM policy in the environment to generate examples for fine-tuning. We perform fine-tuning with LoRA [50]. See App. H for training details.

6.2 Experiment Results

In this subsection, we report the results of different methods, and discuss their implications. Unless otherwise noted, numbers are reported across five runs, for some commercial VLMs such as GPT-o1, we only report one run due to cost consideration.

VLM zero-shot To evaluate the capabilities of state-of-the-art vision-language models, we tested several leading VLMs including LLaVAOneVision [51], Gemini-2.0-flash [11], Gemini-2.0-flash-thinking [11], GPT-4o [10], and GPT-o1 [52], with particular focus on Gemini-2.0-flash-thinking and GPT-o1 as they have demonstrated superior reasoning capabilities across various VLM benchmarks. As shown in Fig. 5, all models achieved notably low success rates on our tasks. While Gemini-2.0-flash-thinking and GPT-o1 showed marginally better performance compared to other models, indicating some improved reasoning capabilities, their performance remains insufficient for solving our complex manipulation tasks. Even the best-performing model, GPT-o1, succeeded in only 15 out of 100 tasks, primarily on simpler cases that did not require sophisticated physical reasoning. This

Table 1. **Post-training performance.** Success rates (%) of post-training variants over the number of iterations.

Method	Iter. 1	Iter. 2	Iter. 3
w/o reflect	58.2	74.4	**77.8**
w/o reflect@test	64.4	76.0	**82.2**
reflect w/ diffusion	66.2	75.8	**82.4**
reflect w/ sim	66.8	75.4	**85.4**

Table 2. **Inference computation cost.** Inference wall clock time per step. MCTS result is averaged over 100 tasks and 1 seed; the others are averaged over 100 tasks and 5 seeds. All experiments are done on a single A100 GPU.

Method	Inference time (s)
Ours w/o reflect@test	0.45
Ours w/ diffusion	11.10
Ours w/ sim	6.05
MCTS	391.42

significant performance gap confirms the necessity of our proposed method for handling physically-grounded reasoning tasks. Detailed evaluation procedures and results can be found in App. J.

MCTS To compare with model-based planning approaches, we implemented a VLM-based MCTS policy. It uses our pretrained VLM policy to generate candidate actions when expanding tree nodes, with heuristic value estimation from the simulator. See App. J for implementation details. As shown in Fig. 5, MCTS achieves a 24.0% success rate—higher than zero-shot VLMs but lower than our method. Notably, while the pretrained VLM policy alone achieves a 47.8% success rate, adding MCTS actually degrades performance. Our analysis revealed that although MCTS helped with some challenging tasks, it would sometimes incorrectly override valid plans from the base VLM policy. We found MCTS to be particularly challenging to tune effectively for our domain for several reasons: (1) it is highly sensitive to value function quality, (2) our tasks require nuanced physical reasoning that is difficult to capture in a value function, and (3) the possibility of succeeding from any state (by clearing the board and starting over) creates minimal value differences between states. These limitations highlight the advantages of our proposed method, which offers a lightweight, flexible approach that requires minimal tuning and can be readily integrated with any VLM policy.

ReflectVLM Our full method outlined in Alg. 1 and 2 incorporates reflection mechanisms in both training and inference phases. To systematically evaluate the impact of reflection, we conducted ablation experiments across several variants. As reported in Fig. 5, the variant without reflection in both training and inference achieved the lowest performance, though it still significantly outperformed the pretrained VLM baseline. The full method using a simulator during inference achieves the highest success rate, serving as an upper bound for our method's performance. When using a diffusion model instead of a simulator during inference, performance degrades slightly. This is unsurprising, as our tasks require nuanced understanding of physics and temporal dynamics—areas where current generative models still face challenges [53, 54]. We expect our method's performance to improve as generative models advance. We also report the post-training dynamics in Table 1. It's observed that the performance of all variants increases as more training is performed and the full method did achieve the highest performance as mentioned above. While the absolute performance gap between variants may appear modest, the additional tasks solved by including reflection are qualitatively significant; see App. A for performance grouped by task difficulty, which demonstrates the significance of the reflection mechanism in solving *hard* tasks. These are typically complex scenarios requiring multiple replanning attempts, such as removing previously placed objects to explore alternative solutions—tasks the pretrained VLM consistently failed to solve. Notably, even without reflection during inference, our method achieves higher success rates than the pretrained baseline. This suggests that the natural language reflection prompts during training help the VLM policy develop better implicit reasoning capabilities. See Fig. 6 in App. B for a representative example, where reflection mechanism iteratively revised suboptimal actions initially proposed by the VLM policy by identifying potentially unfavorable future states. This reflection capability proved crucial for success, as the long-horizon nature of the task required reactive planning and continuous adjustment of the solution strategy. Another point to consider is computation efficiency. Table 2 shows the wall-clock time required per inference step. Compared to MCTS, our method requires only a fraction of the computation time while achieving substantially higher performance, making it particularly appealing as a lightweight and flexible solution for real-world applications.

7 Limitations

In this work, we presented a novel post-training strategy with reflection to improve VLM policies for long-horizon manipulation tasks, demonstrating superior planning capabilities with significantly less compute than traditional approaches like MCTS. Meanwhile, several open problems remain. Our current implementation only uses final outcomes for reflection due to VLM context constraints, hindering fine-grained credit assignment. Future architectures with expanded context windows could enable richer intermediate feedback for more precise action refinement. It is also possible to extend our single-round reflection approach to multiple rounds for iterative refinement while maintaining computational efficiency. Another limitation is the compounding error of the diffusion dynamics model. Incorporating physical constraints and improved architectures could enhance prediction stability over longer horizons. Lastly, our current experiments are limited to a simulated environment. While the tasks are already designed to be extensive, capturing objects of various shapes, sizes, colors, and initial poses, and challenging interlocking structures, an important future step is adapting and validating our reflection mechanism in real-world robotic scenarios. In practice, this would involve human-in-the-loop supervision, where expert interventions provide corrective feedback when the VLM fails. The large-scale simulated data can also be used to augment real-world trajectories in training, thereby mitigating the challenge of collecting real-world data. We believe our method would benefit from continued advances in VLMs and generative models, and we hope it could establish a new foundation with broad applicability to sequential decision-making domains requiring visual understanding, physical reasoning, and long-horizon planning.

References

[1] J. Luo, C. Xu, X. Geng, G. Feng, K. Fang, L. Tan, S. Schaal, and S. Levine. Multistage cable routing through hierarchical imitation learning. *IEEE Transactions on Robotics*, 40:1476–1491, 2024. doi:10.1109/TRO.2024.3353075.

[2] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms, 2020. URL https://arxiv.org/abs/1907.03146.

[3] J. Cui and J. Trinkle. Toward next-generation learned robot manipulation. *Science Robotics*, 6, 2021.

[4] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477, 2011. doi:10.1109/ICRA.2011.5980391.

[5] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning, 2020. URL https://arxiv.org/abs/2010.01083.

[6] D. Driess, J.-S. Ha, and M. Toussaint. Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image, 2020. URL https://arxiv.org/abs/2006.05398.

[7] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Learning compositional models of robot skills for task and motion planning, 2021. URL https://arxiv.org/abs/2006.06444.

[8] X. Chen, J. Dai, X. Li, B. Peng, M. Singh, S. Tao, X. Wang, Y. Wang, Y. Xia, et al. Pali-x: On scaling up a multilingual vision and language model. *arXiv preprint arXiv:2305.18565*, 2023.

[9] J. Bai, S. Bai, S. Du, S. Han, P. Liu, et al. Qwen-vl: A versatile vision-language model for understanding, generation, and retrieval. *arXiv preprint arXiv:2308.12966*, 2023.

[10] OpenAI. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

[11] Google. Introducing gemini: Our largest and most capable ai model. https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/#ceo-message, 2024. Accessed: 2024-02-14.

[12] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning, 2023. URL https://arxiv.org/abs/2304.08485.

[13] D. Driess, A. Black, H. Kataoka, Y. Tsurumine, Y. Koyama, N. Mansard, D. Fox, K. Choromanski, B. Ichter, K. Hausman, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[14] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023. URL https://arxiv.org/abs/2212.06817.

[15] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

[16] L. X. Shi, Z. Hu, T. Z. Zhao, A. Sharma, K. Pertsch, J. Luo, S. Levine, and C. Finn. Yell at your robot: Improving on-the-fly from language corrections, 2024. URL https://arxiv.org/abs/2403.12910.

[17] F. Liu, K. Fang, P. Abbeel, and S. Levine. Moka: Open-world robotic manipulation through mark-based visual prompting, 2024. URL https://arxiv.org/abs/2403.03174.

[18] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh. Physically grounded vision-language models for robotic manipulation, 2024. URL https://arxiv.org/abs/2309.02561.

[19] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Driess, P. Florence, D. Sadigh, L. Guibas, and F. Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities, 2024. URL https://arxiv.org/abs/2401.12168.

[20] J. Huang, X. Chen, S. Mishra, H. S. Zheng, A. W. Yu, X. Song, and D. Zhou. Large language models cannot self-correct reasoning yet, 2024. URL https://arxiv.org/abs/2310.01798.

[21] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023. URL https://arxiv.org/abs/2212.10560.

[22] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

[23] M. Renze and E. Guven. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*, 2024.

[24] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[25] L. Pan, M. Saxon, W. Xu, D. Nathani, X. Wang, and W. Y. Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*, 2023.

[26] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.

[27] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[28] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[29] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[30] K. Cheng, Y. Li, F. Xu, J. Zhang, H. Zhou, and Y. Liu. Vision-language models can self-improve reasoning via reflection, 2024. URL https://arxiv.org/abs/2411.00855.

[31] X. Yu, B. Peng, V. Vajipey, H. Cheng, M. Galley, J. Gao, and Z. Yu. Exact: Teaching ai agents to explore with reflective-mcts and exploratory learning, 2025. URL https://arxiv.org/abs/2410.02052.

[32] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning, 2023. URL https://arxiv.org/abs/2311.17842.

[33] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models, 2023. URL https://arxiv.org/abs/2307.05973.

[34] S. Belkhale, T. Ding, T. Xiao, P. Sermanet, Q. Vuong, J. Tompson, Y. Chebotar, D. Dwibedi, and D. Sadigh. Rt-h: Action hierarchies using language, 2024. URL https://arxiv.org/abs/2403.01823.

[35] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, Q. Vuong, T. Zhang, T.-W. E. Lee, K.-H. Lee, P. Xu, S. Kirmani, Y. Zhu, A. Zeng, K. Hausman, N. Heess, C. Finn, S. Levine, and B. Ichter. Pivot: Iterative visual prompting elicits actionable knowledge for vlms, 2024. URL https://arxiv.org/abs/2402.07872.

[36] N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, and K. Ikeuchi. Gpt-4v (ision) for robotics: Multimodal task planning from human demonstration. *IEEE Robotics and Automation Letters*, 2024.

[37] A. Mei, G.-N. Zhu, H. Zhang, and Z. Gan. Replanvlm: Replanning robotic tasks with visual language models. *IEEE Robotics and Automation Letters*, 2024.

[38] A. Mei, J. Wang, G.-N. Zhu, and Z. Gan. Gamevlm: A decision-making framework for robotic task planning based on visual language models and zero-sum games. *arXiv preprint arXiv:2405.13751*, 2024.

[39] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models, 2023. URL https://arxiv.org/abs/2310.10639.

[40] Y. Du, M. Yang, P. Florence, F. Xia, A. Wahid, B. Ichter, P. Sermanet, T. Yu, P. Abbeel, J. B. Tenenbaum, L. Kaelbling, A. Zeng, and J. Tompson. Video language planning, 2023. URL https://arxiv.org/abs/2310.10625.

[41] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning, 2020. URL https://arxiv.org/abs/1802.08705.

[42] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[43] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083, 2018. URL https://api.semanticscholar.org/CorpusID:52939433.

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

[45] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

[46] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations, 2021. URL https://arxiv.org/abs/2011.13456.

[47] T. Brooks, A. Holynski, and A. A. Efros. Instructpix2pix: Learning to follow image editing instructions. *arXiv preprint arXiv:2211.09800*, 2022.

[48] J. Luo, C. Xu, F. Liu, L. Tan, Z. Lin, J. Wu, P. Abbeel, and S. Levine. Fmb: A functional manipulation benchmark for generalizable robotic learning. *The International Journal of Robotics Research*, 2024.

[49] H. Liu, C. Li, Y. Li, and Y. J. Lee. Improved baselines with visual instruction tuning, 2024. URL https://arxiv.org/abs/2310.03744.

[50] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[51] B. Li, Y. Zhang, D. Guo, R. Zhang, F. Li, H. Zhang, K. Zhang, P. Zhang, Y. Li, Z. Liu, and C. Li. Llava-onevision: Easy visual task transfer, 2024. URL https://arxiv.org/abs/2408.03326.

[52] OpenAI. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

[53] B. Kang, Y. Yue, R. Lu, Z. Lin, Y. Zhao, K. Wang, G. Huang, and J. Feng. How far is video generation from world model: A physical law perspective, 2024. URL https://arxiv.org/abs/2411.02385.

[54] S. Motamed, L. Culp, K. Swersky, P. Jaini, and R. Geirhos. Do generative video models learn physical principles from watching videos?, 2025. URL https://arxiv.org/abs/2501.09038.

[55] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550 (7676):354–359, 2017.

A Performance grouped by task difficulty

We group the tasks by their difficulty and aggregate the performance in different groups to demonstrate the significance of the reflection mechanism in solving *hard* tasks. Specifically, we divide the 100 test tasks into 3 groups (24 easy, 60 medium, and 16 hard tasks) according to the number of steps required to solve these tasks with an oracle policy. Note that the search space grows exponentially as the number of steps increases. As shown in Table 3, although the performance is similar between different methods on medium-level tasks, the success rate of adopting reflection is significantly higher than without reflection on hard tasks.

Table 3. **Post-training performance grouped by task difficulty.** Success rates (%) of post-training variants grouped by the number of steps required to solve a task. Mean of 5 seeds. **Easy**: 0-9 steps, **medium**: 10-14 steps, **hard**: ≥15 steps (max 22 steps).

Method	Easy	Medium	Hard
w/o reflect	83.3	86.3	37.5
w/o reflect@test	85.8	**88.7**	52.5
reflect w/ diffusion	93.3	82.7	65.0
reflect w/ sim	**97.5**	85.0	**68.8**

B Qualitative example

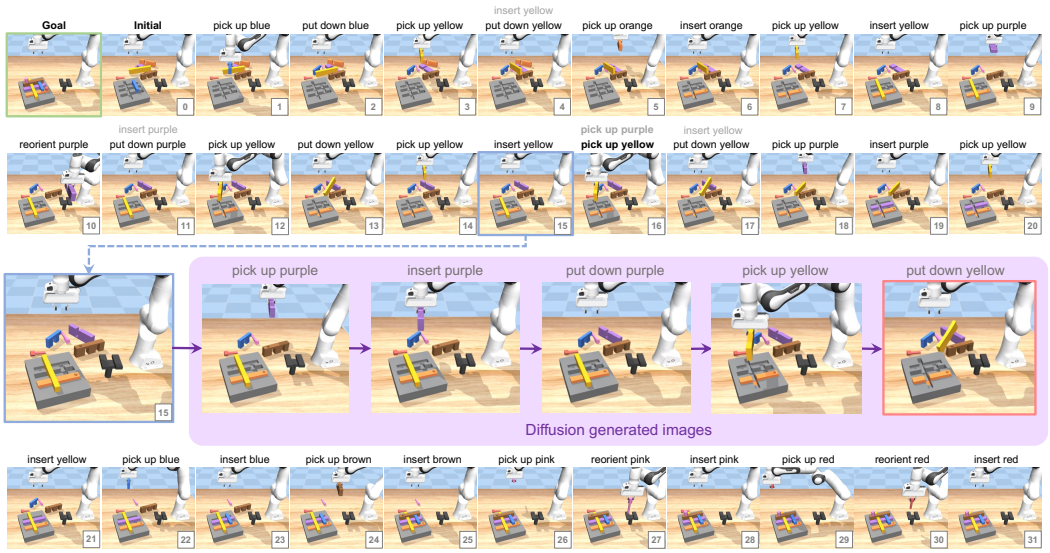

Figure 6. **Filmstrip of our method solving a complicated assembly task.** Frames are indexed by timestep. The goal image is in the top-left corner (with a green border). Each frame is the observation after executing the action (in black) above it. The other action in gray is the original action proposed by the VLM if it is revised after reflection. We highlight the reflection process at timestep 15, where the VLM first proposes an action to pick up the purple brick, but after reflection, it chooses to pick up the yellow brick instead as the generated future state (red-bordered image) shows little progress towards the goal.

C Task generation

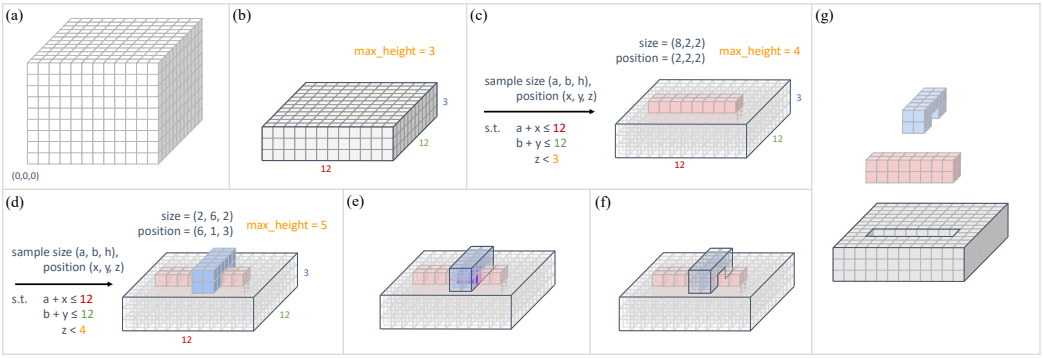

Figure 7. **Example of task generation.** (a) Voxel representation of the board. (b) Generating a base board. (c) Generating a red brick. (d) Generating another blue brick. (e) Critical voxels (highlighted in purple) at the intersection of the two bricks. (f) Handling intersection by assigning the critical voxels to the red brick. (g) Explosion view of the board consisting of three interlocking pieces.

We here describe the procedure to generate assembly boards in detail with an example. A board is discretized into voxels and can be represented by a 3d array, where each value indicates the piece the voxel belongs to. Initially none of the voxels is occupied, so they are all set to an empty value 0, as shown in Fig. 7(a). Then we iteratively add pieces to the board. We first sample the size of the base board, which is (12, 12, 3) in this example (Fig. 7(b)). Then we set these voxels to 1 to indicate they belong to the base board. We also maintain a variable `max_height`, which represents the highest layer that contains non-zero voxels. To generate a brick, we sample its size and position subject to some constraints (Fig. 7(c)). The first two constraints ensure that this brick is within the range of the base board, and the third constraint makes sure this brick will intersect with some previously generated brick. As before, we set the value of the red voxels to 2 to indicate they are from the new brick. Note that the voxels in the lower layer previously have a value of 1 since they belonged to the base board, but now their value is rewritten to 2. This also creates a hole on the base board. After generating this brick, we also update `max_height` to 4 since we have 4 layers now. Fig. 7(d) shows the process of generating another brick. As the new blue brick intersects with the old red brick at the four critical voxels highlighted in purple (Fig. 7(e)), we can assign the value of these critical voxels to either that of the red one or the blue one. For example, keep these voxels to the red brick results in an opening on the blue one (Fig. 7(f)). Stopping the generation process here gives us a board with three interlocking pieces, as shown in Fig. 7(g).

D Samples of generated tasks

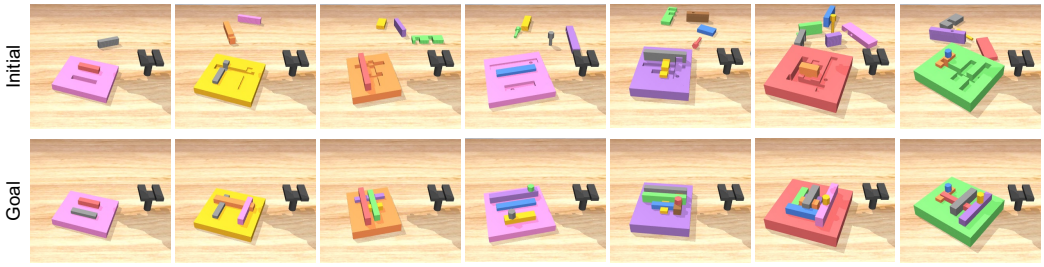

Figure 8. **Samples of generated tasks.** We procedurally generate a variety of multi-stage manipulation tasks, ranging from simple peg insertion to complex assembly tasks that contains multiple interlocking pieces. Top: initial configurations. Bottom: goal configurations.

E Task statistics

We provide the statistics of the 100 evaluation tasks in Table 4.

Table 4. **Experimental task statistics.**

	Min	Max	Median	Average
Number of pieces	5	8	5	5.53
Number of actions required	4	22	13	12.13

F Low-level action primitives

We implemented four action primitives: {`pick up`, `insert`, `reorient`, `put down`}. Specifically, "`pick up`" grasps a piece that is not in hand and picks it up. It can then be inserted into the board using the "`insert`" action, or put back on the table using "`put down`". By invoking "`reorient`", the object in hand can be reoriented with the black fixture if necessary, so that it is in a suitable pose for insertion. Table 5 shows the success rates of each action primitive.

Table 5. **Success rates of action primitives.**

Action primitive	pick up	reorient	insert	put down
Success rate (%)	98.4	90.4	90.1	97.5

G Expert policy

Algorithm 3 Expert Policy

Require: task status $status_{\text{global}}$, object in hand obj_{hand},

if obj_{hand} is not None **then**
  if all predecessors of obj_{hand} are `DONE` **then**
    if obj_{hand} is in `BAD_D` state **then**
      return "reorient obj_{hand}"
    else if obj_{hand} is in `BLOCKED_S` state **then**
      return "put down obj_{hand}"
    else
      return "insert obj_{hand}"
    end if
  else
    return "put down obj_{hand}"
  end if
else
  if $status_{\text{global}}$ == `READY` **then**
    choose an object obj in `READY` or `BAD_D` state
    return "pick up obj"
  else if $status_{\text{global}}$ == `BAD_B` **then**
    choose an object obj in `BAD_B` state
    return "pick up obj"
  else
    return "done"
  end if
end if

The expert policy assumes access to the states of the objects in the simulator, such as the position and orientation of each piece. It is also provided with the dependency graph of the task, as discussed in Sec. 5. We define the status of each piece to be one of the following:

- `DONE`: if it is properly inserted into board;

- `READY`: if it is not inserted yet but ready to be manipulated;
- `BAD_B`: if it is in *bad* state since it is *blocking* other bricks, implying it needs to be removed;
- `BAD_D`: if it is in *bad* state since it is *down*, implying it needs to be reoriented;
- `BLOCKED_P`: if it is *blocked* since some *predecessor* brick(s) should be inserted before;
- `BLOCKED_S`: if it is *blocked* since some *successor* brick(s) is inserted before.

Based on the status of each piece, we can also define a set of possible statuses for the entire assembly task:

- `DONE`: if the board is fully assembled, i.e., all pieces are in `DONE` state;
- `READY`: if some brick is in `READY` or `BAD_D` state;
- `BAD_B`: if we need to reset some brick(s) to proceed as it is blocking other bricks.

When queried, the expert policy first checks the status of each piece according to the simulation states, and decide the status of the whole task based on the statuses of all pieces. Then it decides the action to take following Algorithm 3.

H Training details

H.1 VLM Policy

Architecture. As shown in Fig. 9, the architecture of our VLM consists of a vision encoder and a Large Language Model (LLM). By default, we use `clip-vit-large-patch14-336` [1] as the vision encoder, and `vicuna-13b-v1.5` [2] as the LLM. We initialize our VLM with LLaVA-v1.5 weights [3] that are pre-trained on general visual instruction tuning datasets. Since our task prompts consist of interleaved images and text (refer to Sec. I), we use a shared vision encoder to extract latent embeddings and concatenate them back to an input sequence.

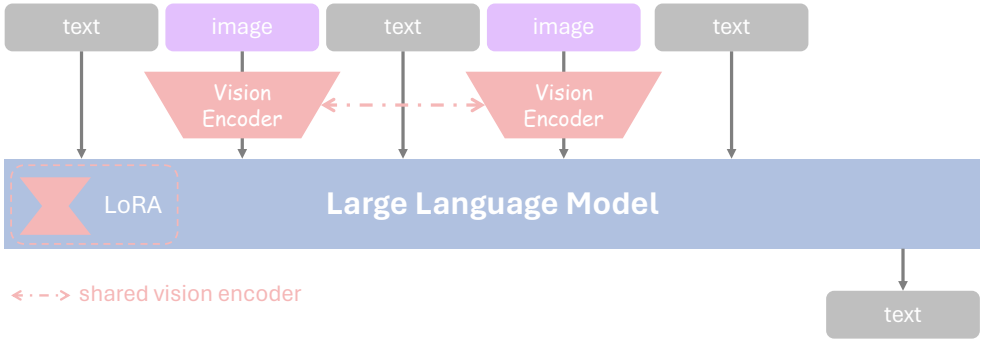

Figure 9. **Architecture of our VLM.** The model consists of a vision encoder and an LLM. We also add Low-Rank Adaptation (LoRA) [50] layers to LLM for efficient adaptation. The input sequence contains interleaved images and text, where images are encoded into latent embeddings with a shared vision encoder. Finally, the concatenation of text and image embeddings are fed into VLM for multimodal reasoning.

Training Parameters. The full training parameters are listed in Table 6. For efficient adaptation of VLM to our task, we only finetune newly added LoRA [50] layers. The rank of LoRA layers is 128 by default.

Table 6. **Training parameters of VLM.**

Res.	LoRA Rank	Training Epoch	Batch Size	Optimizer	Warmup Epoch	Learning rate BC	Learning rate Iter. 1,2,3	Weight Decay	LR Schedule
336px	128	1	128	AdamW	0.03	5e-5	1e-5	0.0	Cosine

[1] https://huggingface.co/openai/clip-vit-large-patch14-336
[2] https://huggingface.co/lmsys/vicuna-13b-v1.5
[3] https://huggingface.co/liuhaotian/llava-v1.5-13b

H.2 Diffusion Dynamics Model

Data Generation. We generate $10K$ different boards and use sub-optimal policies to collect transitions. The sub-optimal policies are implemented by setting a probability $p = \{0.2, 0.5, 0.7, 0.9, 1.0\}$ to replace the expert action by a random action. We collect $50K$ trajectories; each has a maximum length of 50 and is terminated upon success. In total, we have about $1M$ transitions. We randomly sample $50K$ transitions for evaluation, and the rest is used for training.

Training Parameters. The full training parameters are listed in Table 7. We initialize the Diffusion Dynamics Model with pretrained Instructpix2pix [47] [4].

Table 7. **Training parameters of Diffusion Dynamics Models.**

	UNet	Decoder
Resolution	512px	512px
Training steps	20K	4K
Batch size	640	160
Optimizer	AdamW	AdamW
Warmup steps	2K	1K
Learning rate	1e-4	1e-7
Weight decay	0.01	0.01
Beta1, Beta2	0.9, 0.999	0.9, 0.999
Grad norm	1.0	1.0
LR schedule	Cosine	Cosine

I Prompts

I.1 Action proposal prompt

> There is a puzzle consisting of a board and several pieces with different colors on the table. The goal is to assemble the puzzle with the robot arm. In each step, one of the following four actions can be taken: pick up [obj], put down [obj], reorient [obj], and insert [obj], where [obj] refers to the piece to be manipulataed. The image of the goal state is: <image>. The image of the current state is: <image>. The most recently executed actions are: {`history`}. What action should be taken next? Note that [obj] should be a color chosen from the following list: {`colors`}.

I.2 Reflection prompt

> There is a puzzle consisting of a board and several pieces with different colors on the table. The goal is to assemble the puzzle with the robot arm. In each step, one of the following four actions can be taken: pick up [obj], put down [obj], reorient [obj], and insert [obj], where [obj] refers to the piece to be manipulataed. The image of the goal state is: <image>. The image of the current state is: <image>. The most recently executed actions are: {`history`}. The next five steps planned by the model is {`init_plan`}, from which we are going to only execute the first action. Note that if the full plan was executed sequentially, the future state would be: <image>. What action should be taken for the immediate next step? Note that [obj] should be a color chosen from the following list: {`colors`}. You can modify the initial plan if it leads to an undesired future state.

[4] https://huggingface.co/timbrooks/instruct-pix2pix

J Baseline details

J.1 Zero-shot VLMs

We prompt state-of-the-art close-sourced and open-sourced VLMs for zero-shot evaluation, including LLaVA-Onevision, Gemini-2.0 (`gemini-2.0-flash-exp`), Gemini-2.0-thinking (`gemini-2.0-flash-thinking-exp-1219`), GPT-4o and GPT-o1. We resize all input images to 336×336 pixels for fair comparisons with our model. We set the generation temperature and max planing step to 0 and 50. The evaluation prompt is:

> You are an intelligent robot equipped with cameras and robotic arms, your primary task is to observe and interact with the objects on the desktop.
>
> {Action proposal prompt (Sec. I.1)}
>
> You can only output the action, e.g., pick up red. Do not output anything else.

Since the instruction following capability of LLaVA-Onevision is quite limited, we cannot extract valid actions from its response. For other close-sourced VLMs, we list the detailed evaluation results in Table 8. We also visualize some success cases in Figures 10 and 11, and failure cases in Figures 12 to 15.

Table 8. **Detailed evaluation results of zero-shot VLMs.**

Model	Success Trajectory ID / Planning Steps	Max Steps	Min Steps	Avg Steps
Gemini-2.0	5/6, 12/4, 16/18, 47/11, 60/4, 86/6	18	4	8.2
Gemini-2.0-Thinking	5/6, 12/4, 40/20, 47/16, 50/8, 60/8, 86/10, 90/11	20	4	10.4
GPT-4o	12/15, 16/5, 19/4, 47/10, 60/4, 90/6	15	4	7.3
GPT-o1	12/9, 16/6, 17/15, 47/8, 50/16, 58/18, 60/14, 62/33, 66/6, 67/12, 72/32, 77/9, 85/9, 86/6, 90/4	33	4	13.1

J.2 MCTS

We implemented MCTS similar to AlphaGo Zero [55] but with a VLM policy for action proposal and a heuristic value estimator. States and actions are represented by nodes and edges, respectively. The algorithm iteratively expands the search tree and estimates the value for different actions. We store the visit count $N(s,a)$, total action value $W(s,a)$, and action value $Q(s,a) = W(s,a)/N(s,a)$ on edges. Each iteration consists of three phases: (1) select, (2) expand, and (3) backup.

In select phase, it traverses the tree by selecting the edge that has the largest action value $Q(s,a)$ plus an upper confidence bound $U(s,a) = c_{\text{explore}}\sqrt{\sum_{a'} N(s,a')}/(1+N(s,a))$, where c_{explore} is the factor to balance exploring less visited edges and exploiting edges with high value. We use $c_{\text{explore}} = 0.5$ in our experiments. If there is no actions associated to a node yet, it samples 5 top-likelihood actions with the VLM, with duplicates removed, and adds them to the node.

In expand phase, it expands the selected edge by simulating the action in the simulator, getting the next state, and adding the new state to the tree as a new node. It then estimates the value of the new state by rolling out the expert policy from that state. The estimated value is $V = \exp(-\lambda S)$, where S is the number of steps the expert policy takes to reach the goal from the new state, and $\lambda = 0.1$ is a scaling factor.

In backup phase, it updates the statistics of the edges on the path from the root to the expanded node: $N(s,a) \leftarrow N(s,a)+1$, $W(s,a) \leftarrow W(s,a)+V$, and $Q(s,a) \leftarrow W(s,a)/N(s,a)$.

The search completes after 50 iterations. Among all actions connected to the root node, the action with the highest Q value is chosen to execute. We replan with MCTS at each timestep.

Figure 10. **Success cases of zero-shot VLMs.** Top: Gemini-2.0; Middle: Gemini-2.0-Thinking; Bottom: GPT-4o.

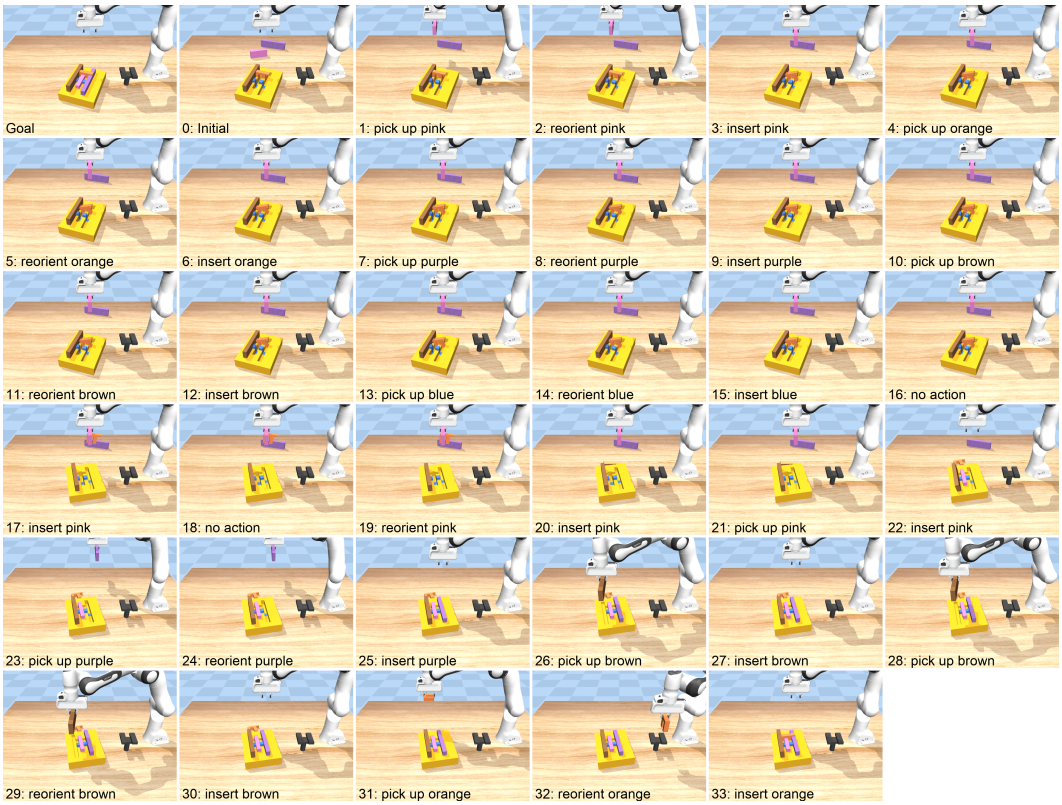

Figure 11. **Success cases of zero-shot VLMs (GPT-o1).**

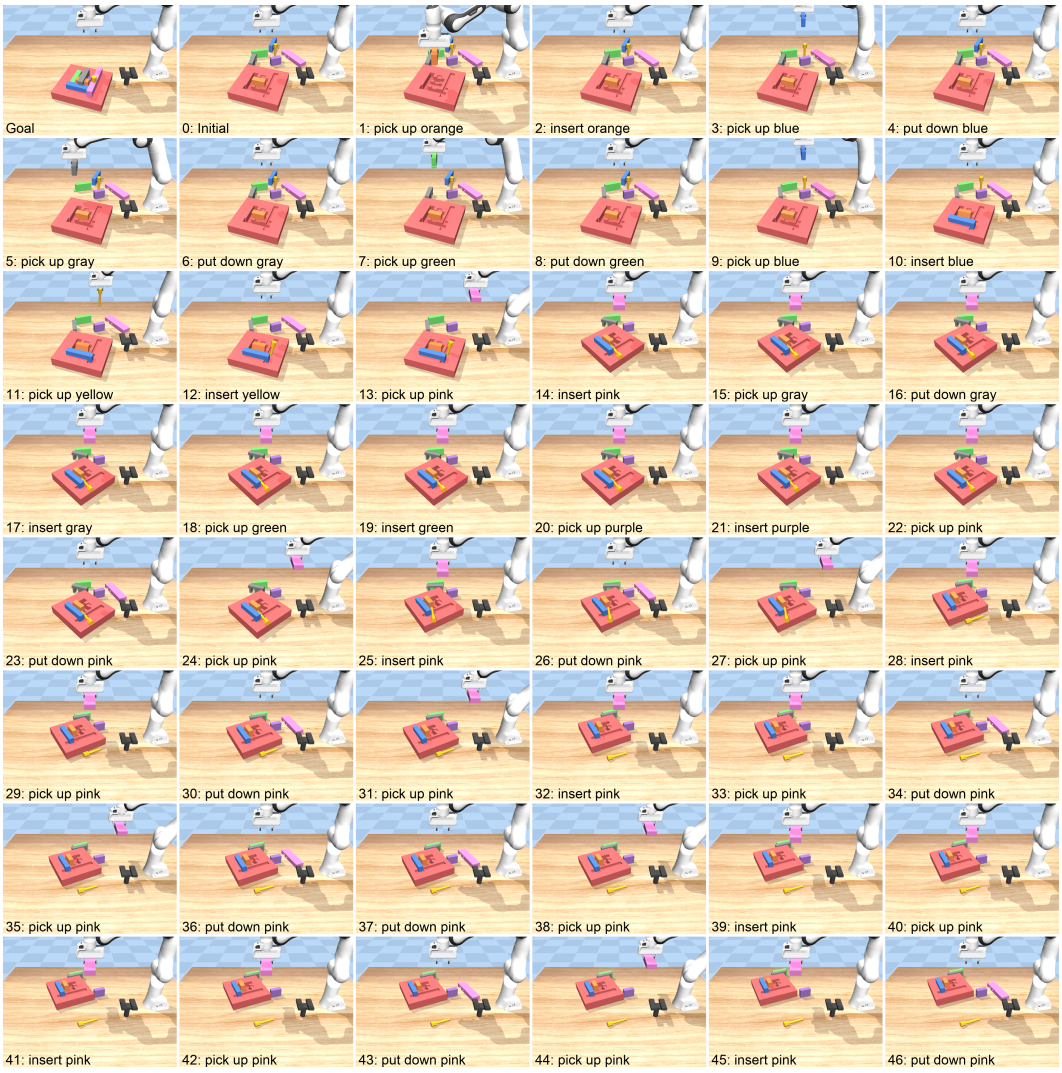

Figure 12. **Failure case of Gemini-2.0.**

Figure 13. **Failure case of Gemini-2.0-Thinking.**

Figure 14. **Failure case of GPT-4o.**

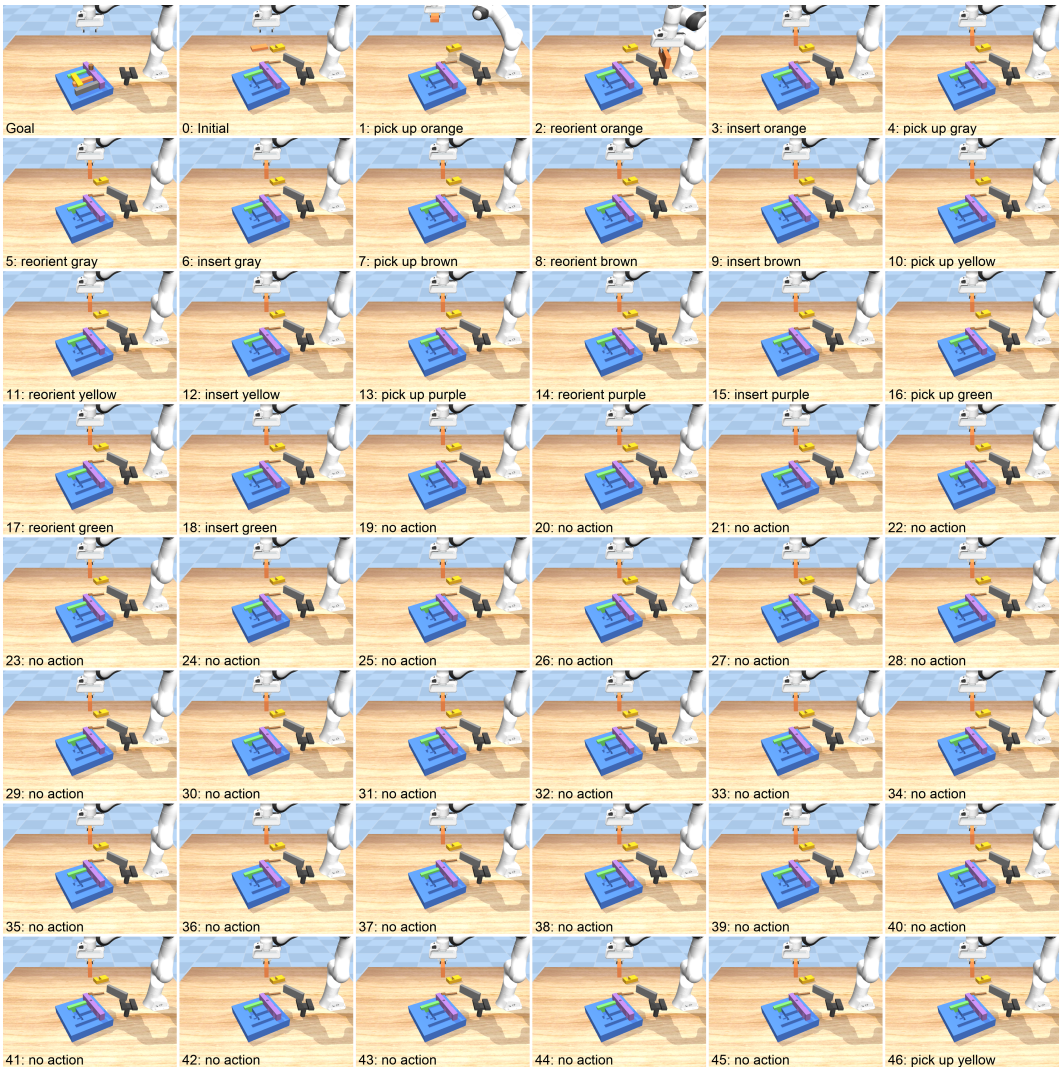

Figure 15. **Failure case of GPT-o1.**

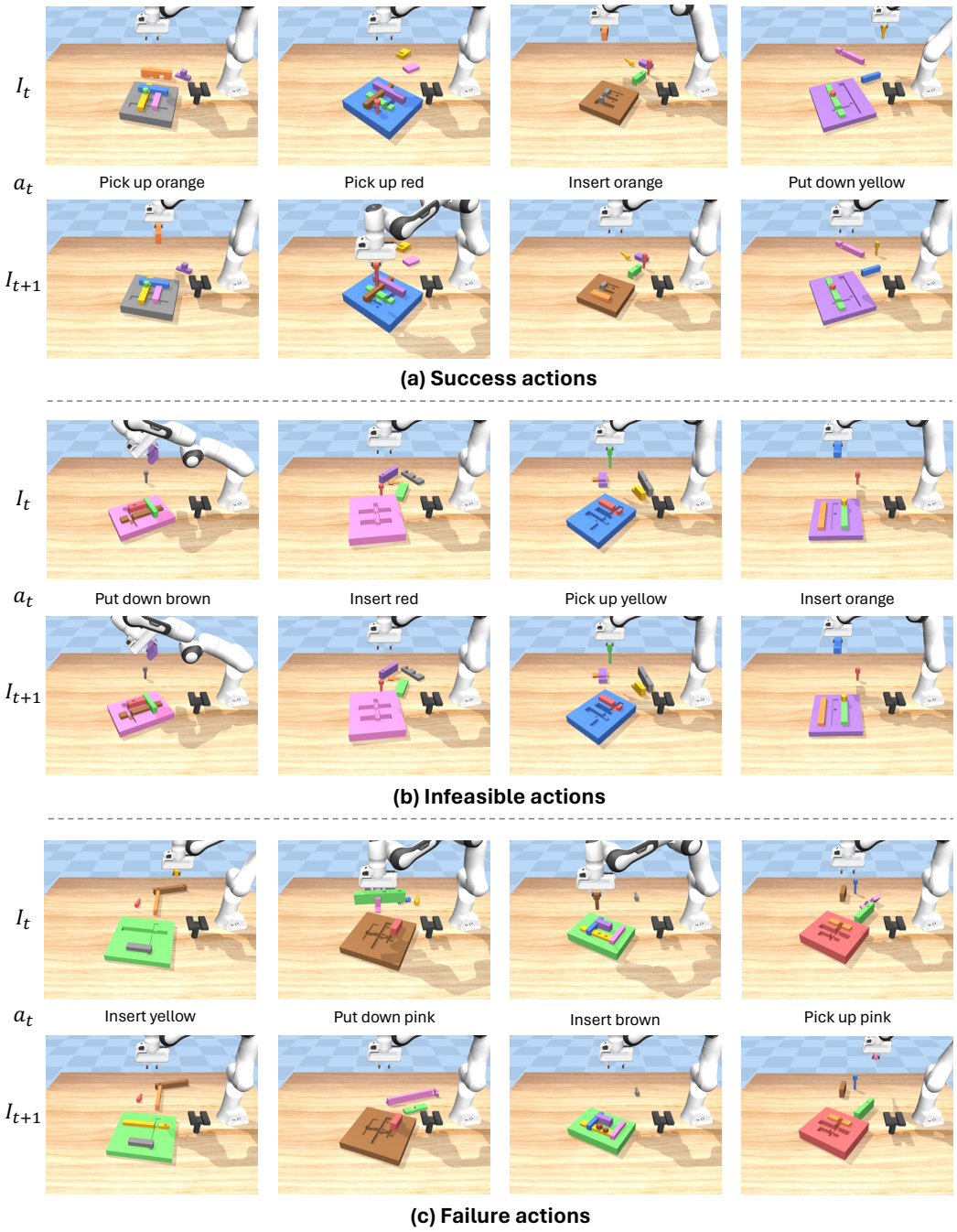

Figure 16. **Examples of Diffusion Dynamic Models.**