VLA-Reasoner: Empowering Vision-Language-Action Models with Reasoning via Online Monte Carlo Tree Search

Wenkai Guo[*,1], **Guanxing Lu**[*,2], **Haoyuan Deng**[1], **Zhenyu Wu**[3], **Yansong Tang**[2], **Ziwei Wang**[†,1]

[1] School of Electrical and Electronic Engineering, Nanyang Technological University
[2] Tsinghua Shenzhen International Graduate School, Tsinghua University
[3] School of Intelligent Engineering and Automation, Beijing University of Posts and Telecommunications
`wenkai001@e.ntu.edu.sg`, `ziwei.wang@ntu.edu.sg`

Abstract— Vision-Language-Action models (VLAs) achieve strong performance in general robotic manipulation tasks by scaling imitation learning. However, existing VLAs are limited to predicting short-sighted next-action, which struggle with long-horizon trajectory tasks due to incremental deviations. To address this problem, we propose a plug-in framework named VLA-Reasoner that effectively empowers off-the-shelf VLAs with the capability of foreseeing future states via test-time scaling. Specifically, VLA-Reasoner samples and rolls out possible action trajectories where involved actions are rationales to generate future states via a world model, which enables VLA-Reasoner to foresee and reason potential outcomes and search for the optimal actions. We further leverage Monte Carlo Tree Search (MCTS) to improve search efficiency in large action spaces, where step-wise VLA predictions seed the root. Meanwhile, we introduce a confidence sampling mechanism based on Kernel Density Estimation (KDE), to enable efficient exploration in MCTS without redundant VLA queries. We evaluate intermediate states in MCTS via an offline reward shaping strategy, to score predicted futures and correct deviations with long-term feedback. We conducted extensive experiments in both simulators and the real world, demonstrating that our proposed VLA-Reasoner achieves significant improvements over the state-of-the-art VLAs. Our method highlights a potential pathway toward scalable test-time computation of robotic manipulation.

I. INTRODUCTION

Vision-Language-Action models (VLAs) [1], [2], [3] leverage the grounded perception and commonsense reasoning of large, pre-trained vision–language models (VLMs) to advance general-purpose robot manipulation. Within a supervised imitation learning paradigm, they map visual observations and natural-language instructions directly to sequences of low-level actions using extensive robot demonstration datasets [4], [5], [6]. By decoupling task specification from policy learning, VLAs adapt to diverse manipulation scenarios via language prompts and image conditions. Recent results show stronger generalization across object categories and environments, and reduced dependence on explicit task engineering, positioning VLAs as a promising route toward scalable embodied intelligence [2], [7].

However, current VLAs also face critical limitations. As the action prediction of VLAs fundamentally relies on direct mappings from short-horizon environment states to actions, they remain fragile during deployment. This short-sighted prediction discards long-horizon sequential dependencies, becoming a primary cause of incremental deviations across
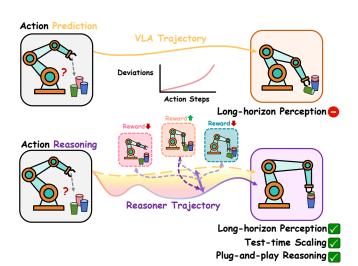

Fig. 1: **VLA-Reasoner** augments VLA models with test-time reasoning via online tree search, enabling more robust and interpretable robotic manipulation than baselines.

tasks and environments. Consequently, the accuracy and exploration capability of VLAs are significantly constrained. This raises a core question: *"Can VLAs explore the long-horizon future influence of actions at test time, and decide the optimal action?"*

To this end, we propose a plug-in framework named VLA-Reasoner that empowers off-the-shelf VLAs with the ability to foresee future states via test-time scaling. Our proposed VLA-Reasoner can effectively mitigate the incremental deviations of VLAs caused by the lack of considering future impact (Figure 1). Specifically, VLA-Reasoner samples and rolls out possible action trajectories to generate future states via a world model, where the future states and corresponding actions can reflect the potential outcomes. To enhance search efficiency, we employ MCTS to handle the expansive action space, in which step-wise VLA predictions seed the root node. We introduce a KDE-based confidence distribution that samples candidates in MCTS from an expert-like prior, reducing redundant VLA queries while preserving exploration. Since sparse task feedback arrives only at episode ending, we design an offline reward shaping strategy to evaluate intermediate states in MCTS, providing dense feedback signals that correct deviations with stable long-horizon guidance. VLA-Reasoner effectively improves the reasoning
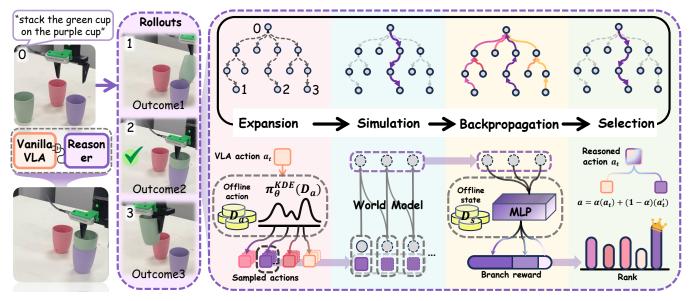
Fig. 2: **The overall pipeline of VLA-Reasoner.** At test time, a lightweight and modified MCTS searches for the optimal action conditioned on the VLA prediction. The search is steered by expert-like sampling and dense reward shaping, which guide expansion and backup throughout the tree. The method is plug-and-play, and it can be attached to any VLA-based manipulation policy and consistently improves performance across tasks, environments, and robot embodiments.

capability of VLAs in long-horizon trajectory tasks, through enabling structured exploration in expansive action spaces and foreseeing the potential outcomes of the current action.

Our method significantly delivers consistent gains in both simulation and on real robots. On the LIBERO benchmark, wrapping a modest baseline VLA with VLA-Reasoner lifts it beyond competing VLAs. In real-world deployments, our approach achieves higher success rates compared to popular VLAs fine-tuned with a few demonstrations, indicating stronger generalization and adaptivity at test time. Our contributions are summarized as follows:

- We propose a plug-in framework named VLA-Reasoner that empowers VLAs with structured reasoning to address their incremental deviations during deployment.
- We adapt a modified test-time MCTS to search efficiently rather than just use it. We apply KDE for efficient plausible expansion, and provide an offline-based reward shaping method to evaluate intermediate states.
- We conduct extensive experiments that validate the effectiveness of integrating MCTS into VLAs for test-time optimization. We also show the potential to achieve great real-world performance with a few data acquisitions.

II. RELATED WORK

Vision-Language-Action Models. As robot learning increasingly demands generalist policies, Vision-Language-Action (VLA) models have emerged as a promising paradigm [1], [8], [2], [3]. Recent efforts integrate Vision-Language Models (VLMs) into VLA systems, leveraging their broad vision-language understanding and reasoning capabilities [9], [10]. Trained on large-scale robotic datasets [4], [6], VLA models achieve state-of-the-art manipulation performance. Their generalization, inherited from VLMs, enables rapid adaptation to novel and unstructured environments [11], [12]. Despite this progress, current VLAs still face challenges. Reliance

on expert data and limited use of VLM reasoning restricts the generalization and transferability of VLA policies across domains. Recent work improves VLAs via low-level designs like data augmentation, action re-ranking, and reinforced fine-tuning [13], [14], [15], but these gains are often incremental and brittle to distribution shifts in deployment. To mitigate the compounding errors in deployment of VLAs, VLA-Reasoner introduces explicit test-time reasoning through model-guided tree search, systematically enhancing both the reasoning ability and adaptability of VLA deployment.

Monte Carlo Tree Search. Monte-Carlo Tree Search (MCTS) is a popular Markov Decision Process (MDP) planning algorithm to explore complex action decision space [16], [17]. MCTS especially serves as a reasoning augment mechanism in deep learning from agent learning in game decision [18], [19] and Large Language Model (LLM) optimization [20], [21]. The Tree Search algorithm performs strong self-guided reasoning ability to solve long and difficult problems, therefore, it plays a heuristic role in our method. MCTS has been adapted in robot control and planning [22], [23], [24]. Prior work mainly targets path planning on discrete graphs (e.g., go), which differs fundamentally from model-driven robotic manipulation with VLAs. In deployment, VLAs should reason over multi-modal observations and contact-rich, non-stationary dynamics, which makes vanilla MCTS difficult to run online. We therefore adapt MCTS for robotic manipulation by leveraging efficient simulated interactions, rather than using it directly.

Model-based Planning. Classical model-based planning simulates a dynamics model forward and selects the action sequence with the lowest cost (or highest reward). Trajectory-optimization and MPC variants have been widely used in robotics with analytic or calibrated models [25], [26]. With learned dynamics, "world models" replace physics engines

and enable planning directly from images—e.g., latent-space video models and visual-foresight methods that predict future frames for goal-conditioned manipulation [27], [28]. More recently, sequence models plan by proposing and scoring full trajectories (e.g., Trajectory Transformer, Diffuser) [29], [30]. Our setting departs from these in two key aspects. First, we attach planning at test time to a pretrained VLA: the VLA's one-step prediction serves as the root, and a learned world model is used solely for online look-ahead rather than policy training. Second, instead of continuous trajectory optimization, we adopt discrete tree search guided by a data-driven action prior and dense visual rewards, thereby avoiding hand-tuned costs and task-specific heuristics.

III. METHOD

In this section, we first show the pipeline of our framework as Figure 2, and then present the formulation of our work (Section III-A). We adapt MCTS (Section III-B) for efficient test-time expansion and backpropagation on the VLA prediction without disturbing real world execution. Under MCTS manner, we apply KDE to efficiently sample action candidates for exploration rather than repeatedly querying VLA (Section III-C). We also shape the reward based on offline data with a simple method to densely evaluate intermediate candidate-driven states (Section III-D).

A. Problem Statement

VLAs aim to generalize robot manipulation by mapping multimodal inputs (states from the environment o_t, language instructions of the task l) to actions a_t^{VLA}. The prediction of pretrained VLA can be formulated as $a_t^{VLA} = \pi_\theta(o_t, l)$, where θ means the parameters of the model and is frozen. Since the prediction relies only on the current state o_t, the absence of future consideration leads to deviations that progressively grow over time.

Our goal is to mitigate this deviation during test time, which enables foreseeing the future impact of current action. Instead of directly deploying VLA, we simulate future states using a world model $\mathcal{W}$ and score them with a reward r. At timestamp t, we feed action a_t and state o_t into $\mathcal{W}$ and get feedback of next state o_{t+1}, this process can be represented as $\mathcal{W}(o_{t+1} \mid a_t, o_t)$. After simulations under a MCTS manner, VLA-Reasoner rolls out an action $a_t^{Reasoner}$. We then inject a_t with VLA-Reasoner:

$$a_t = \alpha(a_t^{\text{VLA}}) + (1 - \alpha)(a_t^{\text{Reasoner}}) \tag{1}$$

where α represents the injection strength to control the balance of two actions. The a_t^{Reasoner} integrates future impact into the decision, providing long-horizon guidance, whereas the final action a_t represents the executed decision during deployment.

B. Online Monte Carlo Tree Search

The key to VLA-Reasoner lies in leveraging a tree structure consist of possible action trajectories and corresponding states for guided and directional search, where reasoning is enabled with interaction in a world model. We follow the MCTS manner for efficient tree search, and adapt MCTS to a simple

implementation in test time. At each step t, VLA-Reasoner proceeds through four steps: **(a) Expansion**, **(b) Simulation**, **(c) Backpropagation**, and **(d) Selection**. A node in MCTS process with index i is denoted as the state o_i, and the reward of this node is denoted as r_i. The corresponding action to generate the node is a_i. We further introduce the specific implementation of each step as follows.

a) Expansion: The expansion step aims to expand the selected node o_i (initially the root node, the superscripts from here to Selection are slightly obfuscated to make them intuitive), to generate its children nodes. As actions are directly related to the generation of new states, we sample a set of actions $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ from a distribution π_θ under a Top-k strategy. For action a_i, the expansion can be formulated as:

$$\begin{aligned}\textbf{Sample:} \quad & \tilde{\mathcal{A}}_i = \{a^{(n)}\}_{n=1}^N \sim \pi_\theta, \\ \textbf{Top-}k\textbf{:} \quad & \mathcal{A}_i^{\text{Top-k}} = \argmin_{A \subseteq \tilde{\mathcal{A}}_i, |A|=k} \sum_{a \in A} \|a - a_i\|_2\end{aligned} \tag{2}$$

where we get the $\mathcal{A}_i^{\text{Top-k}}$ as the candidates to expand from a randomly large sample set $\tilde{\mathcal{A}}_i$, which are closest to the a_i in Euclidean distance (k here is relatively small compared to N).

b) Simulation: To explicitly evaluate the influence of these sampled actions, we simulate future states using a learned action-aware world model [31] as the backbone to predict visual states conditioned on actions. The simulation formulates:

$$o_{i+1} = \mathcal{W}(a_i, o_i) \tag{3}$$

where the world model rolls out the next state o_{i+1} under a given action a_i and state o_i. We embed robot actions into its latent space and finetune it on a small robot dataset, aligning multimodal inputs for plausible transition generation.

c) Backpropagation: As the tree search meets the truncating condition, MCTS updates the node's value in bottom-up order, from each leaf node to the root. The overall value of node o_i is $\mathcal{Q}(o_i)$, which is balanced by combining the cost of the visit count $N(a_i)$. The value and visit count are updated:

$$\begin{aligned}N(o_i) &= \sum_{a \in \mathcal{A}_i^{\text{Top-k}}} N(a), \\ \mathcal{Q}(o_i) &= \frac{N(o_i)\, r_i + \sum_{j \in \text{children}} N(o_j)\, \mathcal{Q}(o_j)}{N(o_i) + \sum_{j \in \text{children}} N(o_j)}\end{aligned} \tag{4}$$

where r_i is the reward and $children$ are the candidates of expansion. To efficiently deploy with low latency, visit counts are limited as they are the main inference cost. To overcome this, we estimate the visit count of a sampled action (instead of the state o_i) by its probability density naturally derived from the distribution (Section III-C). The reward r_j is introduced in the following (Section III-D).

d) Selection: This step aims to select the preferred node that balances search quality and efficiency. The selection follows two factors: the value $\mathcal{Q}(o_i)$ and the visiting counts $N(o_i)$ of the node o_i. We adopt the Upper Confidence Bound (UCB) strategy [32]. The selection follows the formula:

$$a_{selected} = \arg\max_{i \in k} \mathcal{Q}(o_{i+1}) + c \cdot \sqrt{\frac{\ln N(\hat{o}_i)}{1 + N(o_i)}} \quad (5)$$

where c is a constant to constrain the exploration (e.g., $\frac{1}{\sqrt{2}}$), $\hat{o}_i$ stands for the parent node of o_i. The UCB strategy enables a balance between exploration and exploitation during search. The node is selected with the highest UCB score.

These four steps are repeated in a round of iteration, where it takes real state and action as input. The whole process constructs an independent Monte Carlo Tree of current robot states as we use a world model to dictate the transitions. Generating such a tree structure represents finding an optimized search space of actions, in which we can rollout the best candidate. Then inject this candidate into the VLA prediction following Equation (1) As VLA can always predict next-best action, we sparsely conduct MCTS on VLA generations, where the efficiency can be apparently optimized while action quality is improved.

C. Distribution for Efficient Sampling

To efficiently sample actions for expansion, we utilize Kernel Density Estimation (KDE) to model the distribution of actions from offline data. KDE is a non-parametric way to estimate the probability density function of a random variable, which allows us to generate diverse action candidates that are likely to be effective based on historical data. With a dataset of actions $\{a_1, a_2, \ldots, a_n\}$, the KDE can be formulated as:

$$\pi_\theta^{\text{KDE}}(a) = \frac{1}{N} \sum_{i=1}^{N} K_h(a - a_i), \quad (6)$$

where K is the kernel function (we use a Gaussian kernel here), and h is the bandwidth parameter that controls the smoothness of the estimated density, θ means the above involved hyperparameters.

We can then efficiently sample the distribution via $a_i \sim \pi_\theta^{\text{KDE}}(\cdot)$. Since KDE returns a probability density $p_i = \pi_\theta^{\text{KDE}}(a_i)$, we can treat it as a Monte-Carlo estimate of how often the action (and its corresponding state) would be visited under large-scale sampling. In practice, this gives a soft prior for the visit count as $N(a) \propto p(a)$, which is crucial for efficient backpropagation.

For policies that output action chunks (e.g., a sequence of actions), we adapt the KDE to sample action chunks. We treat each chunk as a single entity and apply KDE to the entire chunk, allowing us to sample diverse and effective action sequences. To avoid long chunks washing out fine-grained corrections and affecting KDE performance, we modify the used policy to roll out in a controllable size of actions (e.g., below 8) while still benefiting from sequence-level proposals.

Algorithm 1 VLA-Reasoner

Preparation : world model $\mathcal{W}$, KDE prior π_θ^{KDE}, reward network MLP.
Input : VLA proposal a_t^{VLA}, current observation o_t
Output : final action a_t

1 **Init:** Create root node $o^{(0)} \leftarrow o_t, a^{(0)} \leftarrow a_t^{\text{VLA}}$.;
2 **for** *depth* $d = 0$ **to** $MaxDepth$ **do**
3   // Expansion
    $\mathcal{A}^{\text{Top-k}} \leftarrow \text{TopK}(\pi_\theta^{\text{KDE}}, a^d)$ ▷ *Equation* (2)
4   **for** $a \in \mathcal{A}^{\text{Top-k}}$ **do**
5   // Simulation
        $\hat{o} \leftarrow \mathcal{W}(o^{(d)}, a)$
6   // Evaluation
        $\hat{r} \leftarrow \text{MLP}(\hat{o})$
7   // Backpropagation
        $\text{Update}(o^{(d)}, \hat{r})$ ▷ *Equation* (4)
8   **end**
    // Selection
9     $o^{(d)} \leftarrow \text{SelectUCB}(o^{(d)})$ ▷ *Equation* (5)
10     $a^{(d)} \leftarrow a(o^{(d)})$
11 **end**
    // Reasoner rollout
12 $a_t^{\text{Reasoner}} \leftarrow \arg\max_{a_i, i \in children} \mathcal{Q}(o_i)$
    // Action injection with strength α
13 $a_t \leftarrow \alpha \cdot a_t^{\text{VLA}} + (1 - \alpha) \cdot a_t^{\text{Reasoner}}$

D. Vision-based Reward Shaping

Rewards can reflect task progress. To evaluate the reward of model-generated states, we follow the idea that changes in visual observations are a key indicator of progress [33]. A straightforward approach is to measure this variation and derive rewards directly from image frames. However, manipulation trajectories are usually dense. To reduce redundancy and preserve meaningful changes, we down-sample image sequences from the offline dataset. This ensures that consecutive frames contain sufficient variation while retaining most progress information. Based on the down-sampled sequence, we assign relative ground-truth rewards through linear interpolation between 0 and 1. For example, in a 10-frame sequence, the 5th frame is assigned a reward of $\frac{5}{9}$.

For efficient and consistent scoring, we utilize the ResNet-34 [34] with frozen ImageNet-pretrained weights as the visual encoding backbone, followed by a 2-layer MLP training with MSE loss. The training objective can be formulated as:

$$\psi^\star = \arg\min_\psi \mathcal{L}_{\text{MSE}}(\text{MLP}(o_t), \{r_t\}) \quad (7)$$

where the reward is generated with $r_t = \text{MLP}(o_t)$. With a simple but efficient design (the network takes less than 30 minutes to train), our design enables accurate reflection to boost the performance (Section IV-C).

Pseudocode for VLA-Reasoner is provided above (Algorithm 1). Components except the world model are trained on the same dataset used to finetune the VLA. For the world model, we additionally collect a small set of failure demonstrations to finetune it for predicting failure cases.

TABLE I: **Results in Simulations.** Average success rates across 500 episodes for LIBERO and 100 episodes for SimplerEnv. Our method outperforms OpenVLA-SFT on all 4 direction tasks and Octo-Small/SpatialVLA on 4 tasks. Bold entries mark the highest success rates, underlined for second-best. Asterisked results are chosen baselines and locally evaluated for fairness.

LIBERO	Spatial	Goal	Object	Long	Average
Diffusion Policy [37]	78.3%	68.3%	**92.5%**	50.5%	72.6%
GRAPE [38]	87.6%	<u>82.2%</u>	<u>91.2%</u>	<u>55.8%</u>	<u>79.2%</u>
TraceVLA [39]	84.6%	75.1%	85.2%	54.1%	74.8%
SpatialVLA [40]	88.2%	78.6%	89.9%	55.5%	78.1%
*OpenVLA-SFT [1]	84.6%	79.2%	86.6%	53.4%	76.0%
+VLA-Reasoner	**91.2%**	**82.4%**	90.6%	**59.8%**	**81.0%**
SimplerEnv	Block	Spoon	Carrot	Eggplant	Average
*Octo-Small[3]	5%	43%	10%	48%	26.5%
+VLA-Reasoner	**13%**	**50%**	**32%**	**54%**	**37.3%**
*SpatialVLA	23%	18%	22%	73%	34.0%
+VLA-Reasoner	**29%**	**35%**	**28%**	**75%**	**41.8%**

IV. EXPERIMENTS

In this section, we present both simulation and real-world experiments to explore the following key questions:

1) **Test-time gains.** How does VLA-Reasoner enhance the performance of different pretrained VLAs across diverse environments during the test time?
2) **Real-world Applicability.** How does VLA-Reasoner help generalist VLAs in real-world manipulation tasks?
3) **Robustness.** Can VLA-Reasoner adapt to varied settings while maintaining performance?

For **Q1**, we conduct experiments in 2 simulation environment (LIBERO [35] and SimplerEnv [36]) with 8 specific tasks based on 3 popular general robot policies. For **Q2**, we evaluate the practicality of the VLA-Reasoner on 5 challenging real-world tasks on top of 2 advanced VLAs. For **Q3**, we ablate two main factors (one supporting our method and the other one supporting VLA deployment) to assess adaptation and stability. We also conduct ablation on specific technique designs to test the effectiveness.

A. VLA-Reasoner in Simulation

a) Experiment Setup: We evaluate our method with a combination of 3 popular generalist robot policies: OpenVLA, which is a VLA model with a parameter size of 7B [1]; Octo-Small [3], which is a transformer combining a diffusion head for action prediction, with a parameter size of 27M; SpatialVLA, which is a spatial-enhanced VLA model with a parameter size of 4B [40]. We follow the architecture of iVideoGPT [31] to train an action-aware world model, with a parameter size of 600M. All the training phases (including KDE estimation and reward shaping) rely on the same public datasets generated from the simulators. For the world model, we additionally supplement its training with a small set of failure demonstrations collected from the rollouts of the pretrained VLA itself, enabling the model to capture those failures during deployment. For execution, OpenVLA and Octo roll 1 action every time, and SpatialVLA

rolls 4 actions as a chunk every time. We choose LIBERO [35] and SimplerEnv [36] for simulation. For LIBERO, we utilize 4 task suites: Spatial, Object, Goal, and Long. Each suite contains 500 expert demonstrations distributed across 10 language-conditioned tasks with 50 variations, designed to evaluate policy generalization across different spatial configurations, object types, goals, and long-horizon task sequences. In SimplerEnv, we employ 4 representative tasks on the WidowX robot: 1. **Block**: *Stack green block on yellow block*; 2. **Spoon**: *Put spoon on towel*; 3. **Carrot**: *Put carrot on plate*; 4. **Eggplant**: *Put eggplant in yellow basket*. We use OpenVLA-SFT to refer to the OpenVLA model finetuned on the public LIBERO dataset. All the training processes are conducted on a server with 6 NVIDIA RTX 6000 GPUs.

b) Quantitative Study: Experiment results are shown in Table I. As the success rate is the primary metric of evaluation in two benchmarks, our method improves the absolute task-set performance on OpenVLA-SFT by 5% on average, the reasoner also improves the task performance on Octo-Small by 9.8%, and SpatialVLA by 7.8%, on average. It is noticeable that compared to those variants developed from OpenVLA, our plug-and-play method can directly improve the performance of the backbone to the state-of-the-art level without large-scale and skillful post-training, which is required for those variants. When compared to SpatialVLA, which is augmented with better spatial understanding capability, our method outperforms it in all task suites. Moreover, our method shows notable improvements on inherently difficult tasks such as stack block, which typically require precise sequential reasoning and are sensitive to slight prediction deviations. By introducing step-wise simulated inference, VLA-Reasoner enables more deliberate future planning, reducing incremental deviations and significantly boosting task success.

c) Qualitative Study: We attribute the gains to test-time tree search that dynamically spawns counterfactual branches, allowing flexible exploration of alternative futures without
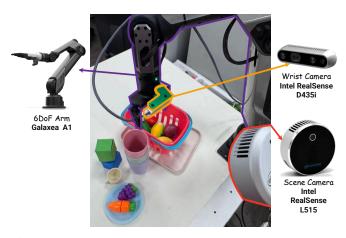
Fig. 3: **Setup of real world experiments.** We conduct diverse tasks in the real world to identify the limitations of current VLAs and validate our method.

TABLE II: **Results in Real World.** Average success rates of 5 tasks in different scenarios. Each task is evaluated 20 times. Our method apparently improves OpenVLA and $\pi_0 - $FAST in all tasks.

Method	Block	Fruit	1 Cup
OpenVLA	25%	45%	20%
+VLA-Reasoner	**40%**	**70%**	**40%**
$\pi_0 - $ FAST [41]	70%	80%	60%
+VLA-Reasoner	**80%**	**90%**	**75%**
Method	**2 Cups**	**Circle**	**Overall**
OpenVLA	5%	15%	22%
+VLA-Reasoner	**15%**	**40%**	**41%**
$\pi_0 - $ FAST	40%	70%	64%
+VLA-Reasoner	**50%**	**75%**	**74%**

disturbing real execution. As our method enables getting future possible deviations caused by current actions, the backpropagation offers a look-ahead evaluation, bringing the Markovian deployment with a longer horizon to mitigate the limitation. To support efficient deployment of VLA-Reasoner, we adopt the mentioned techniques and finetuned the world model with the same data size as finetuning VLA, to reduce computational cost and reliance on the dataset. Detailed analysis is provided in the ablation study.

B. Deployment in Real-world Environment

a) Experiment Setup: To evaluate the performance of the VLA-Reasoner in the real world with real robots. We design and test 5 real-world tasks via deploying our method on two cutting-edge VLAs: an open-sourced popular model OpenVLA-7B [1], and an advanced commercial model $\pi_0 - $ FAST [41] with local finetuning. The training phases use the same datasets, and we collect 10 failure cases for each task to supplement the training of the world model. After every inference, OpenVLA predicts the next 1 action and $\pi_0 - $FAST predicts the next 5 actions as a chunk. We evaluate with a fixed Galaxea-A1 robot arm. Images are provided by one side-fixed camera and one wrist-fixed camera (the arm-fixed camera is not used for OpenVLA since it does not support wrist image input). Real-world inference is conducted on an NVIDIA RTX 4090 GPU. The setup is visualized in Figure 3.

For each evaluation task, we collect 20 demonstrations with the initial positions of the primary objects slightly randomized. We design 5 tasks that test physical awareness, manipulation precision, and long-horizon understanding. The tasks include:

1) **Block**: *Stack the green cube on the blue cube.*
2) **Fruit**: *Pick the grape and place the grape into the basket.*
3) **1 Cup**: *Stack the green cup on the purple cup.*
4) **2 Cups**: *Stack the green cup and the pink cup on the purple cup.*
5) **Circle**: *Circle around the cup.*

b) Results: Experiment results are presented in Table II. In real-world experiments, our method can significantly improve finetuned VLA models during deployment. It averages an improvement of OpenVLA with an absolute gain of 19%, a relative gain of 86.4%, as the baseline shows a poor performance of 22% success rate. It can also surprisingly boost π_0-FAST with absolute gain of 10%, relative gain of 15.6%. Besides the strengths shown in Section IV-A, we find that injecting a directional future-conditioned feedback to action can improve the awareness of current execution, and thus avoid failure modes. As VLAs struggle to succeed in the real world due to environment shifting and the embodiment gap, we analyze a specific case to reveal the short-sighted incremental deviations during real-world deployment, and showcase how our approach mitigates these kinds of deviations without disturbing action execution.

c) Case Study: Humans can easily handle those relatively difficult manipulations because we naturally reason over space and time, predicting how each move will change the future. The same ability should be built into robotic manipulation so the policy looks ahead, not just the present. Without such foresight, policies tend to produce unstable control and suboptimal decisions in fine-grained tasks. We examine a representative challenge: *Stack the green cup on the purple cup.*, which requires precise object localization, adaptive grasping, and fine-grained motion planning under visual distraction. As shown in Figure 4 (top), the baseline $\pi_0 - $ FAST fails to properly align with the target due to the deviations from accumulative immature movement. In contrast, VLA-Reasoner performs test-time simulation of future possible cases, enabling proactive correction before execution. As illustrated in Figure 4 (bottom), the reasoning phase conducted in the world model reflects the most possible future states that may happen via the rewards (annotated on the lower left corner), and the action candidate with highest reward is chosen to inject into the VLA rollout action, the robot adjusts its waypoint mid-trajectory and achieves a more stable, deliberate manipulation. This case reveals how VLA-Reasoner empowers VLAs with reasoning ability during their deployment.
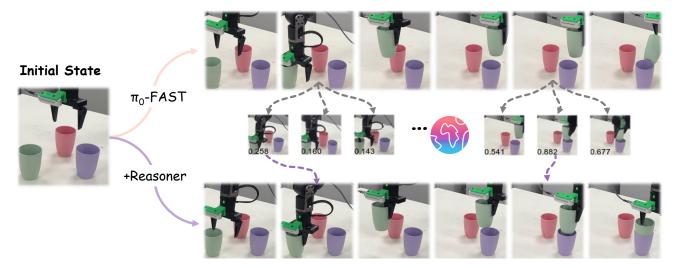
Fig. 4: **Case Visualization.** The baseline policy (π_0-FAST, top row) suffers from excessive action drift and fails by such deviations. With reasoning, VLA-Reasoner (bottom row) proactively corrects misalignment via reward-guided search, enabling success.
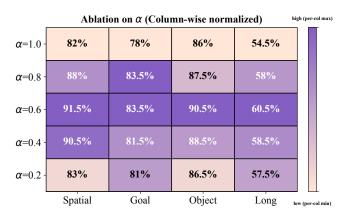

Fig. 5: **Analysis on injection strength α.** α controls the trade-off between the VLA action and the reasoner action; a larger α assigns greater weight to the VLA action. $\alpha = 1.0$ means the vanilla VLA.

C. Ablation Analysis

This section aims to evaluate the robustness and sensitivity of VLA-Reasoner under different injection strengths, and to validate whether its outstanding performance gains arise from our designs rather than other possible solutions. We conduct controlled ablations on LIBERO-Spatial. Two key factors are studied: (1) the *Injection Strength* factor α to control the mix of VLA rollouts and sampled actions, and (2) the main techniques, including *Sampling Criteria* and *Reward Shaping*. The base model remains identical to the main experiments.

a) Choices of Injection Strength: Figure 5 shows that injecting a reward-based action by MCTS into VLA rollout boosts success performance under all 4 settings. $\alpha = 0.6$ shows the highest success rate in all task suites, which is chosen as the hyperparameter for the main evaluation reported in Table I and Tab. II. Interestingly, the deployment gains most with a moderate strength ($\alpha = 0.6$ and $\alpha = 0.4$), which further reveals that current VLAs trained with limited scale show generalist manipulation ability to a certain extent, but still gain incremental deviations during deployment. These results emphasize the importance of optimizing VLA training in a scalable path or an efficient way, and post-processing of
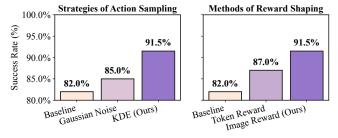

Fig. 6: **Analysis on Techniques.** The comparison validates the design in our method, which is reflected by the significant growth in the success rate.

VLA rollouts to make them more generalizable and intelligent.

b) Choices of Involved Techniques: As mentioned above, applying KDE on offline data enables plausible and efficient sampling, and the reward shaping enables intermediate and relatively accurate reward estimation for feedback. To emphasize their applicability and advantages, we compare the KDE sampling with noisy sampling, which adds a Gaussian noise to VLA rollouts to obtain new actions with similar efficiency. The results are shown in the left part of Figure 6, where the KDE offers plausible choices as they implicitly reflect the smoothed expert behaviors. We compare the image-based reward shaping with the token-based reward head introduced in iVideoGPT [31]. For the token-based reward head, the tokens are directly mapped to a preset reward similar to Section III-D. The results are shown in the right part of Figure 6, where the image-wise reward can better reflect explicit task progress with a simpler implementation.

V. CONCLUSION

We identified a core limitation of current short-sighted VLA deployment and introduced VLA-Reasoner, a plug-in framework that injects test-time reasoning into off-the-shelf VLAs, to mitigate the incremental deviations in deployment. By rolling out imagined futures with a pretrained world model and performing state-based MCTS guided by a KDE action prior and an offline reward desige, VLA-Reasoner reflects

long-horizon consequences back into the current decision. This design optimizes action selection without disturbing the execution or retraining the underlying VLA, and yields consistent gains across simulation and real-world settings with minimal additional supervision.

As the current structure still requires training before deployment to adapt to the specific task, we also see several promising directions under this framework. For example, world models with better fidelity would improve imagined state trajectories and thus the quality of feedback. Second, more principled reward designs, including data-driven and learning based methods, could further stabilize search.

These facets are orthogonal to our main contribution, as VLA-Reasoner remains a compelling paradigm for manipulation. We expect future work to build on VLA-Reasoner and explore scalable test-time computation for general-purpose robotic manipulation.

REFERENCES

[1] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[2] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn *et al.*, "π_0: A vision-language-action flow model for general robot control," 2024. [Online]. Available: https://arxiv.org/abs/2410.24164

[3] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 2024.

[4] O. X.-E. Collaboration, A. O'Neill, A. Rehman *et al.*, "Open X-Embodiment: Robotic Learning Datasets and RT-X Models," Jun. 2024.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar *et al.*, "Rt-1: Robotics transformer for real-world control at scale," in *arXiv preprint arXiv:2212.06817*, 2022.

[6] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du *et al.*, "Bridgedata v2: A dataset for robot learning at scale," in *Conference on Robot Learning (CoRL)*. PMLR, 2023, pp. 1723–1736.

[7] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," 2025. [Online]. Available: https://arxiv.org/abs/2504.16054

[8] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *arXiv preprint arXiv:2307.15818*, 2023.

[9] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[10] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge *et al.*, "Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution," *arXiv preprint arXiv:2409.12191*, 2024.

[11] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu *et al.*, "Vision-language foundation models as effective robot imitators," *arXiv preprint arXiv:2311.01378*, 2023.

[12] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, S. Kirmani, B. Zitkovich, F. Xia *et al.*, "Open-world object manipulation using pre-trained vision-language models," *arXiv preprint arXiv:2303.00905*, 2023.

[13] Z. Xue, S. Deng, Z. Chen, Y. Wang, Z. Yuan, and H. Xu, "Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning," *arXiv preprint arXiv:2502.16932*, 2025.

[14] M. Nakamoto, O. Mees, A. Kumar, and S. Levine, "Steering your generalists: Improving robotic foundation models via value guidance," *arXiv preprint arXiv:2410.13816*, 2024.

[15] G. Lu, W. Guo, C. Zhang, Y. Zhou, H. Jiang, Z. Gao, Y. Tang, and Z. Wang, "Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning," *arXiv preprint arXiv:2505.18719*, 2025.

[16] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk, "Monte carlo tree search: A review of recent modifications and applications," *Artificial Intelligence Review*, vol. 56, no. 3, pp. 2497–2562, 2023.

[17] Y. Pitanov, A. Skrynnik, A. Andreychuk, K. Yakovlev, and A. Panov, "Monte-carlo tree search for multi-agent pathfinding: Preliminary results," in *International Conference on Hybrid Artificial Intelligence Systems*. Springer, 2023, pp. 649–660.

[18] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[19] D. Perez, S. Samothrakis, and S. Lucas, "Knowledge-based fast evolutionary mcts for general video game playing," in *2014 IEEE Conference on Computational Intelligence and Games*. IEEE, 2014, pp. 1–8.

[20] H. Yao, J. Huang, W. Wu, J. Zhang, Y. Wang, S. Liu, Y. Wang, Y. Song, H. Feng, L. Shen *et al.*, "Mulberry: Empowering mllm with o1-like reasoning and reflection via collective monte carlo tree search," *arXiv preprint arXiv:2412.18319*, 2024.

[21] C. Snell, J. Lee, K. Xu, and A. Kumar, "Scaling llm test-time compute optimally can be more effective than scaling model parameters," *arXiv preprint arXiv:2408.03314*, 2024.

[22] H. Vagadia, M. Chopra, A. Barnawal, T. Banerjee, S. Tuli, S. Chakraborty, and R. Paul, "Phyplan: Compositional and adaptive physical task reasoning with physics-informed skill networks for robot manipulators," *arXiv preprint arXiv:2402.15767*, 2024.

[23] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *International Journal of Robotics Research (IJRR)*, 2019.

[24] S. Leisiazar, E. J. Park, A. Lim, and M. Chen, "An mcts-drl based obstacle and occlusion avoidance methodology in robotic follow-ahead applications," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 221–228.

[25] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 4906–4913.

[26] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *Proceedings of International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1714–1721.

[27] D. Ha and J. Schmidhuber, "Recurrent World Models Facilitate Policy Evolution," Sep. 2018.

[28] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[29] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[30] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *arXiv preprint arXiv:2205.09991*, 2022.

[31] J. Wu, S. Yin, N. Feng, X. He, D. Li, J. Hao, and M. Long, "ivideogpt: Interactive videogpts are scalable world models," vol. 37, pp. 68 082–68 119, 2024.

[32] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[33] Y. J. Ma, J. Hejna, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani, P. Xu, D. Driess, T. Xiao *et al.*, "Vision language models are in-context value learners," in *Proceedings of International Conference on Learning Representations (ICLR)*, 2024.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[35] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," vol. 36, pp. 44 776–44 791, 2023.

[36] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani *et al.*, "Evaluating real-world robot manipulation policies in simulation," *arXiv preprint arXiv:2405.05941*, 2024.

[37] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *International Journal of Robotics Research (IJRR)*, 2024.

[38] Z. Zhang, K. Zheng, Z. Chen, J. Jang, Y. Li, S. Han, C. Wang, M. Ding, D. Fox, and H. Yao, "Grape: Generalizing robot policy via preference alignment," *arXiv preprint arXiv:2411.19309*, 2024.

[39] R. Zheng, Y. Liang, S. Huang, J. Gao, H. Daumé III, A. Kolobov, F. Huang, and J. Yang, "Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies," *arXiv preprint arXiv:2412.10345*, 2024.

[40] D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, J. Gu, B. Zhao, D. Wang *et al.*, "Spatialvla: Exploring spatial representations for visual-language-action model," *arXiv preprint arXiv:2501.15830*, 2025.

[41] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine, "Fast: Efficient action tokenization for vision-language-action models," *arXiv preprint arXiv:2501.09747*, 2025.