SITCOM: Scaling Inference-Time COMpute for VLAs

Ayudh Saxena[*] **Harsh Shah**[*] **Sandeep Routray**[*] **Rishi Rajesh Shah**[*] **Esha Pahwa**[*]
Carnegie Mellon University
`{ayudhs, hshah2, sroutra2, rishisha, epahwa}@cs.cmu.edu`

Abstract

Learning robust robotic control policies remains a major challenge due to the high cost of collecting labeled data, limited generalization to unseen environments, and difficulties in planning over long horizons. While Vision–Language–Action (VLA) models offer a promising solution by grounding natural language instructions into single-step control commands, they often lack mechanisms for lookahead and struggle with compounding errors in dynamic tasks. In this project, we introduce *Scaling Inference-Time COMpute for VLAs* (**SITCOM**), a framework that augments any pretrained VLA with model-based rollouts and reward-based trajectory selection, inspired by Model Predictive Control algorithm. SITCOM leverages a learned dynamics model to simulate multi-step action rollouts to select the best candidate plan for real-world execution, transforming one-shot VLAs into robust long-horizon planners. We develop an efficient transformer-based dynamics model trained on large-scale BridgeV2 data and fine-tuned on SIMPLER environments to bridge the Real2Sim gap, and score candidate rollouts using rewards from simulator. Through comprehensive evaluation across multiple tasks and settings in the SIMPLER environment, we demonstrate that SITCOM when combined with a good reward function can significantly improve task completion rate from 48% to 72% using trained dynamics model.

1 Introduction

Robot learning has long been constrained by the need for extensive, labeled data to train effective control policies [1]. Traditional approaches often require meticulously annotated datasets with precise action labels, which are costly and time-consuming to acquire, especially for complex robotic tasks. Furthermore, reliance on task-specific datasets limits the generalization of learned policies to unseen environments, embodiments, or variations of the original task. Although reinforcement learning (RL) offers a potential solution, it exhibits poor sample efficiency for real-world problems and often requires significant–and sometimes unsafe–interaction with the environment. World models [2, 3], which learn predictive representations of the environment to improve sample efficiency and enable risk-free exploration, have emerged as a promising alternative. However, they still struggle to accurately model the diversity and complexity of real-world scenarios, limiting their effectiveness in open-ended tasks.

Vision–Language–Action (VLA) models have recently achieved remarkable success in interpreting natural language instructions and generating corresponding single-step control commands for robotic systems [4, 5, 6]. Despite these advances, deploying VLAs in real-world robotics remains fraught with challenges: they often lack mechanisms for lookahead, struggle to recover from compounding errors, and cannot plan over long horizons in dynamic environments. Such limitations manifest in failures during multi-step tasks like sequential object manipulation, assembly, or navigation in cluttered scenes.

[*]Everyone Contributed Equally – Ordering decided by rolling a dice

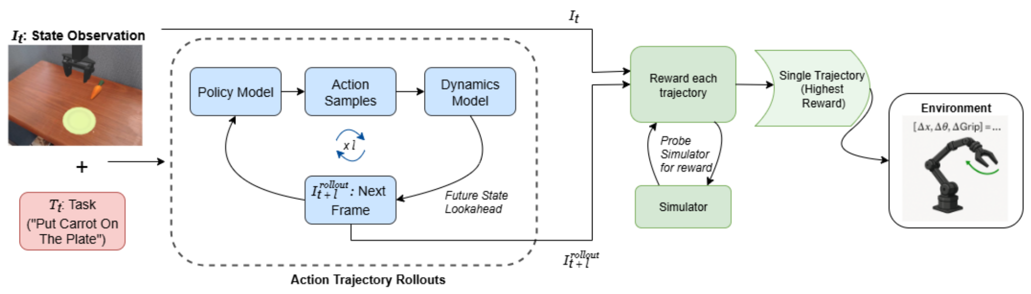

I_t
I_t: State Observation
+
T_t: Task ("Put Carrot On The Plate")
Policy Model
Action Samples
Dynamics Model
x l
$I_{t+l}^{rollout}$: Next Frame
Future State Lookahead
Action Trajectory Rollouts
$I_{t+l}^{rollout}$
Reward each trajectory
Probe Simulator for reward
Simulator
Single Trajectory (Highest Reward)
Environment
$[\Delta x, \Delta\theta, \Delta \mathrm{Grip}] = ...$

Figure 1: Method Diagram explaining the overall SITCOM architecture. Given an initial frame and goal in text description, a VLA predicts actions which is fed into the dynamics model to get the next frame. This process is then iteratively repeated with the output of dynamics model to generate action sequence rollouts. Finally, a reward model is used to rank the action sequences and pick the best one among them to execute in the real-world.

In this work, we introduce the *Scaling of Inference-Time Compute for VLAs* (**SITCOM**) framework (Figure 1), which endows any pretrained VLA with model-based rollout and reward-ranking capabilities inspired by Model Predictive Control (MPC) [7]. At each decision step, SITCOM's VLA policy proposes multiple candidate actions; a learned dynamics model then simulates resulting next states, and this process repeats to generate full multi-step trajectory rollouts. Finally, a reward model ranks these candidate sequences, selecting the trajectory with maximum reward for real-world execution. By performing this *inference-time planning*, SITCOM transforms one-shot VLAs into robust long-horizon planners, improving both reliability and success rates in complex tasks.

In this paper,

- We propose the SITCOM framework, a general-purpose inference-time planning framework that enhances any VLA by simulating multi-step action rollouts and selecting optimal action sequences through a reward mechanism.
- We introduce an efficient transformer-based dynamics model pre-trained on large-scale BridgeV2 [8] data and fine-tuned on SIMPLER environment [9] trajectories to mitigate the Real2Sim gap. To further address error accumulation during long-horizon rollouts, we introduce a DAgger-inspired [10] adaptation strategy that bridges the distributional shift between model inputs and predicted outputs.
- We provide insights on performance gains by rollouts of VLA over number action sequences and the depth of future predictions.

Finally, we validate our approach through comprehensive evaluations on multiple tasks and settings in the SIMPLER environment [9], demonstrating that SITCOM consistently outperforms strong baselines and enables robust long-horizon planning for robotic manipulation.

2 Related Work and Background

Related Datasets. Recent VLA research has drawn on a wide spectrum of datasets, from early human-teleoperated collections (MIME [11], RoboTurk [12]) and scripted large-scale efforts (RoboNet [13], MT-Opt [14]) to more recent real-world corpora (BC-Z [15], RT-1-Kitchen [16], OXE [17], DROID [18]). These resources span hundreds of tasks across diverse robots but differ widely in annotation granularity (raw controls vs. language), complicating unified VLA training. In this work, we build on BridgeData V2 [8] as our primary pretraining corpus: 60k episodes of human and scripted trajectories on a WidowX-250 arm, covering 13 manipulation skills across 24 scenes with over 100 objects, paired with RGB–D, segmentation, and language goal annotations. Its scale, diversity, and grounding make it an ideal foundation for pretraining our Transformer-based dynamics model and fine-tuning OpenVLA within the SITCOM framework.

Vision-language models for robot generalization. Recent advances in vision–language models (VLMs) have expanded robotic generalization by providing visuo-linguistic representations [19, 20], image generation [21], and multimodal reasoning [22, 23, 24], enabling applications such as goal generation [25], reward shaping [26, 27], and representation learning [28, 29]. Vision–language–action models (VLAs) [30, 31, 4] build directly on these advances, achieving state-of-the-art generalist control and strong transfer to novel objects and scenes, though often with limited use of the reasoning capabilities of pre-trained VLMs. Recent work has sought to address this via Chain-of-Thought (CoT) supervision [32, 33], explicitly teaching task decomposition through curated datasets. In contrast, SITCOM takes a complementary path: rather than explicit step-by-step reasoning, we leverage simulator-based rollouts and reward-driven selection to implicitly evaluate futures, guiding long-horizon behavior without requiring curated decompositions.

World models for planning and control. Predictive models of environment dynamics have long underpinned robotics and reinforcement learning [34, 35, 36]. Recent work shows that forecasting future states in pixels [37, 38, 39, 40] or latents [41, 42] can improve sample efficiency and planning: pixel models offer strong visual grounding but are computationally heavy, while latent models are efficient but often task-specific [43, 44]. Large-scale generative video models [45, 46, 47] push realism further but rely on expensive diffusion backbones and language prompts, limiting their use for fine-grained control. In SITCOM, we take a middle ground: an efficient transformer-based dynamics model that predicts future frames in pixel space, lightweight enough for multi-step rollouts. To ensure long-horizon reliability, we adapt it with a DAgger-style [10] strategy that fine-tunes on its own predictions, reducing distributional drift. Pretraining on BridgeV2 and adapting to SIMPLER tasks yields robust, scalable inference-time planning without costly diffusion architectures or dense task supervision.

Reward models for robotic planning. Recent work on reward modeling leverages large pre-trained models to evaluate outputs, with the "LLM-as-a-Judge" paradigm [48] inspiring implementations such as JudgeLM [49] and Generative Judge [50]. While these efforts largely target language evaluation, robotics has adapted similar ideas to visual domains, using language models for reward design [51] or preference tuning [52]. In SITCOM, we instead probe future simulator states to compute rewards: though unrealistic in real-world deployment, this strategy yields interpretable signals and highlights common success and failure modes of VLAs, enabling effective rollout selection during planning.

3 Method

Notation: Let $\mathcal{I}$ denote image observations, $\mathcal{T}$ is the task instructions, $\pi_{\text{VLA}}(\mathcal{I}, \mathcal{T})$ is the VLA model that outputs an action a, $f_{\text{dyn}}(\mathcal{I}, a)$ is the dynamics model, $r(\mathcal{I}_0, \mathcal{I}_l, \mathcal{T})$ is the reward model, n is the number of trajectories, and l is the rollout length.

We propose an inference algorithm for decision-making in an environment using Vision-Language-Action (VLA) models. At each inference step, the agent is provided with an observation $\mathcal{I}$ (an image) and a task instruction $\mathcal{T}$. The VLA model takes $(\mathcal{I}, \mathcal{T})$ as input and outputs an action. To encourage exploration, we set a high sampling temperature and sample n candidate actions.

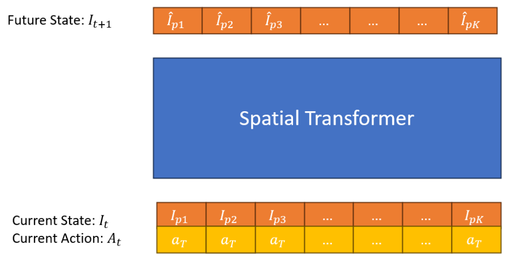
Each sampled action initializes a trajectory. A learned dynamics model, distinct from the real environment, is used to perform rollouts: given an image $\mathcal{I}$ and an action a, the dynamics model predicts the next image $\mathcal{I}'$. Subsequent actions are also temperature-sampeld from the VLA model.

Each trajectory is simulated for l steps and we do this for k trajectories. To perform this simulation, we propose two methods - 1) SITCOM (EnvSim) that uses another instance of the environment to perform the rollouts, 2) SITCOM (Dynamics) where we used a trained dynamics model for the rollouts.

After simulation, we give reward based on the final state of the trajectories (at the pre-defined depth). Our reward design incorporates the gripper-object gap, object-destination distance, and grasp success indicators. The trajectory with the highest reward score is selected, and the trajectory is executed in the real environment. This procedure is repeated at the environment's replanning frequency until task success or termination. Algorithm 1 demonstrates the entire pipeline. Details regarding architecture of dynamics model can be found in section 3.1.

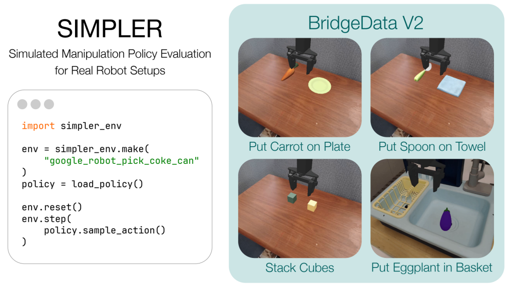
3.1 Training Dynamics Model

We train a dynamics model $f_{dyn}(.)$ that predicts the next state given the current state and action. The model uses an encoder-decoder architecture: the encoder processes image patches, concatenates them



Future State: I_{t+1}
$\hat{I}_{p1}$ $\hat{I}_{p2}$ $\hat{I}_{p3}$ ... ... $\hat{I}_{pK}$
Spatial Transformer
Current State: I_t
Current Action: A_t
I_{p1} I_{p2} I_{p3} ... ... I_{pK}
a_T a_T a_T ... ... a_T

Figure 2: Dynamics Model Architecture



```
import simpler_env

env = simpler_env.make(
    "google_robot_pick_coke_can"
)
policy = load_policy()

env.reset()
env.step(
    policy.sample_action()
)
```

Figure 3: SIMPLER environment and four different tasks with WidowX arm

Algorithm 1: Inference with VLA and Dynamics Model

```
Input: Initial observation $\mathcal{I}_0$, task instruction $\mathcal{T}$
while task not completed and not terminated do
    Sample $n$ actions $\{a_0^1, \dots, a_0^n\}$ from $\pi_{\text{VLA}}(\mathcal{I}_0, \mathcal{T})$ with high temperature;
    foreach $i \in \{1, \dots, n\}$ do
        Initialize trajectory with $\mathcal{I}_0^i \leftarrow \mathcal{I}_0$;
        Apply action $a_0^i$ using dynamics model: $\mathcal{I}_1^i \leftarrow f_{\text{dyn}}(\mathcal{I}_0^i, a_0^i)$;
        for $t = 1$ to $l - 1$ do
            $a_t^i \leftarrow \arg\max_a \pi_{\text{VLA}}(\mathcal{I}_t^i, \mathcal{T})$;
            $\mathcal{I}_{t+1}^i \leftarrow f_{\text{dyn}}(\mathcal{I}_t^i, a_t^i)$;
        end
    end
    Compute rewards $r^i = r(\mathcal{I}_0, \mathcal{I}_l^i, \mathcal{T})$ for each $i$;
    Select trajectory $i^* = \arg\max_i r^i$;
    Execute trajectory $\{a_0^{i^*}, a_1^{i^*}, a_2^{i^*}, \dots a_{l-1}^{i^*}\}$ in the real environment;
    Observe next environment state $\mathcal{I}_0$;
end
```

with action information, and the decoder predicts patches for the next state (Figure 2). To match inference conditions where the model rolls out autoregressively for l steps, we train it to predict future states from its own predictions for l_{train} steps (detailed in Section 5).

The model is trained using combined L1 pixel-wise loss and Learned Perceptual Image Patch Similarity (LPIPS) loss. L1 loss ensures pixel-level accuracy, while LPIPS loss promotes perceptual realism by measuring similarity in deep feature space. This combination balances low-level precision with high-level visual coherence.

We leverage approximately 25,000 trajectories from BridgeV2 [8] for pretraining, providing diverse manipulation scenarios that enable generalizable dynamics learning. To address the Real2Sim gap, we fine-tune on in-domain SIMPLER trajectories, adapting the model to the specific visual appearances and physics of our evaluation environment. This two-stage training approach improves robustness for long-horizon rollouts in the SITCOM framework.

3.2 Finetuning VLA

We train the vision-language-action (VLA) model using a standard cross-entropy loss over discretized action tokens. Since no publicly available expert trajectory data existed for the SIMPLER environment and pretrained real-world models performed poorly in simulation (as shown in the table), we curated our own dataset of 100 expert trajectories. Our dataset curation process involved three steps: first, we ran a pre-trained model [53] to generate initial trajectories; second, we applied heuristic rules to identify successful executions; and finally, we used human filtering to ensure high-quality expert demonstrations. These curated trajectories provide the VLA model with robust examples of successful task execution, enabling it to learn effective mappings from vision-language inputs to low-level control outputs.

	Complete Success				Partial Success			
	Bridge Tasks							
Task	**RT1-X**	**Octo-base**	**Octo-small**	**OpenVLA**	**RT1-X**	**Octo-base**	**Octo-small**	**OpenVLA**
Put spoon on tablecloth	0.042	0.111	**0.486**	0.000	0.125	0.375	**0.833**	0.083
Put carrot on plate	0.083	**0.097**	**0.097**	0.000	0.250	**0.472**	0.264	0.167
Stack green block on yellow block	0.000	0.000	0.014	**0.042**	0.125	0.333	**0.347**	0.125
Put eggplant in basket	0.000	0.417	**0.556**	0.000	0.000	0.639	**0.861**	0.042

Table 1: Performance of multimodal VLA models on different tasks, showing both complete success and partial success rates. Best performance for each task and metric is highlighted in **bold**.

4 Experiment

4.1 Task Setup

To benchmark the effectiveness of our method, we use SIMPLER [9], a suite of open-source simulated environments designed to evaluate generalist robot manipulation policies in a scalable and reproducible manner. We evaluate our model on four tasks (Figure 3) using the 7-DOF WidowX robotic arm. Since SIMPLER does not provide fine-tuning trajectories, we collect 100 multi-task trajectories using successful rollouts from an open source VLA model trained on BridgeData V2 [8], while ensuring diverse object orientations and positions.

We evaluate baseline and our framework across four tasks for WidowX arm and five tasks for Google robot. We focus on following metrics to access perfromance.

Average Success Rate is our main metric that serves as a performance metric for evaluating our models. This metric quantifies the fraction of tasks successfully completed by a policy across diverse scenarios, calculated as:

$$\text{Average Success} = \frac{\text{\# Successful Trials}}{\text{Total Trials}}$$

Partial Success Rate captures instances where the robot achieves part of the goal. For example, the robot could be grasping an object but failing to place it correctly. This metric is important because many robotic tasks involve sequential steps and analyzing partial success helps identify failure points and areas for improvement.

Time ($\propto$ **Compute**) captures the compute required to take a single action. This becomes an important factor for our evaluation, since we propose to scale test-time compute through rollouts. The number of rollouts is limited by the computation budget.

4.2 Baselines

We use the following three leading pretrained architectures as our baselines: **OpenVLA** [4], a 7B-parameter transformer that extends large-scale VLMs to action prediction; **RT-1-X**, an extension of the Robotics Transformer framework scaled to the full Open-X dataset for improved generalization across manipulation skills [5]; and **Octo** [6], a diffusion-policy transformer (97M parameters) designed for cross-environment transfer.

We benchmark multiple vision-language-action models in the SIMPLER [9] simulation environment (Table 1) and identify the Real2Sim gap as a key bottleneck limiting performance. This gap arises from discrepancies between real-world inputs and simulated environments, leading to degraded transferability of policies. To study this systematically, we focus on OpenVLA [4] as our primary baseline, due to its popularity, open-source availability, and comprehensive documentation. To mitigate the Real2Sim gap, we fine-tune OpenVLA on in-domain simulated trajectories, improving its adaptation to the SIMPLER environment.

Initially, we evaluate our approach using an *oracle simulator*, leveraging direct access to ground-truth environment states to benchmark improvements from fine-tuning and dynamics-based rollouts in a controlled setting. Building on this foundation, we then integrate our trained dynamics model and memory-based general reward model, enabling simulation of multi-step rollouts and scoring of candidate trajectories entirely from learned models. This transition allows us to move beyond oracle supervision toward a fully self-contained, scalable simulator framework, setting the stage for broader deployment across diverse robotic tasks.

Tasks	OpenVLA	OpenVLA-SFT	SITCOM (EnvSim)	SITCOM (World Model)
Put Carrot on Plate	0.0	0.50	0.71	0.66
Put Spoon on Table Cloth	0.0	0.63	0.83	0.83
Stack Green Block on Yellow Block	0.042	0.17	0.58	0.62
Put Eggplant in the Basket	0.0	0.63	0.92	0.79
Overall (Avg)	0.01	0.48	**0.76**	0.72

Table 2: Success rates for different methods across robotic manipulation tasks. SITCOM results shown for configuration with rollout length=10, candidates=5

Model	FID (↓)	OFL (↓)
BM. (BridgeV2)	17.0	1.665
FT. Model	**11.2**	**0.992**

Table 3: Comparison of base model (BM) and fine-tuned (FT) model on FID and OFL metrics on transitions extracted from SIMPLER environment. Fine-tuning on in-domain SIMPLER trajectories improves both metrics, demonstrating better prediction of realistic and temporally coherent future state images.

Candidates	1	5	10	15	20	25
Time (s)	21	35	75	100	130	160

Table 4: Planning time (in seconds) for different numbers of candidates

5 Results and Analysis

All reported success rates are computed using reward signals obtained from the simulator (hence assuming perfect knowledge of the environment). We also ablate the effect of predicting future states using either the learned dynamics model or the oracle simulator to isolate sources of error.

5.1 Overall Performance Comparison

We first present the overall performance of our SITCOM framework compared to baseline methods in Table 2. For our main experiments, we use the following SITCOM configuration: rollout length of 10 steps and 5 candidate rollouts. The results demonstrate that SITCOM (EnvSim) achieves the highest performance across all tasks, with SITCOM (World Model) performing comparably. Both variants significantly outperform the baseline methods, OpenVLA and OpenVLA-SFT.

As shown in Table 2, SITCOM (EnvSim) achieves an average success rate of 76%, while SITCOM (World Model) achieves 72%, both dramatically outperforming the baseline OpenVLA (1%) and OpenVLA-SFT (48%) models. This demonstrates the effectiveness of our simulation-guided approach for improving robotic manipulation performance. In the following sections, we analyze how varying these configuration parameters affects performance. Next, we perform various experiments for both SITCOM-EnvSim and SITCOM-Dynamics model, and also conduct ablation studies for the dynamics and reward models.

5.2 Analysis of rollout parameters

Breadth: Number of rollout candidates. First, we demonstrate that scaling the number of rollout candidates is beneficial for robotic tasks. We observe continuous gains until 25 candidates for some tasks, while for other tasks the gains begin to saturate before, as shown in Figure 4. We also report the time taken for varying candidates in Table 4.

While we notice that time increases with scaling the number of candidates, the entire pipeline supports parallelism, allowing this time to be controlled. We elaborate on this in the future work section.

Depth: Length of Rollouts. Next, we vary the rollout lengths for the SITCOM-Dynamics model. The optimal rollout length may vary across tasks, as some tasks require more in-depth planning while others benefit from short-term planning. While we plan to develop a method to estimate the optimal rollout length for each task, we currently keep it fixed and experiment with various rollout lengths.

We find that for challenging tasks such as "Put Eggplant in Basket," our model benefits from longer rollouts, as demonstrated in Figure 5. The results show that performance improves as we increase the rollout length, with particular gains observed in complex manipulation tasks.

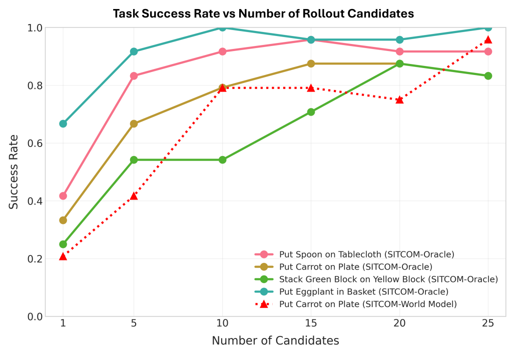


Figure 4: Scaling with number of rollouts for SITCOM-EnvSim and SITCOM-Dynamics models.

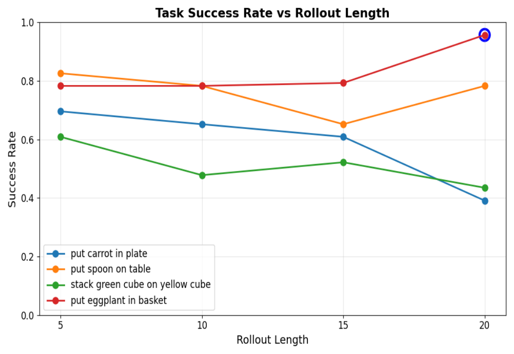


Figure 5: Scaling with varying rollout lengths. The performance of the dynamics model decreases with the increase in rollout length, explaining the drop in overall performance for some of the tasks.

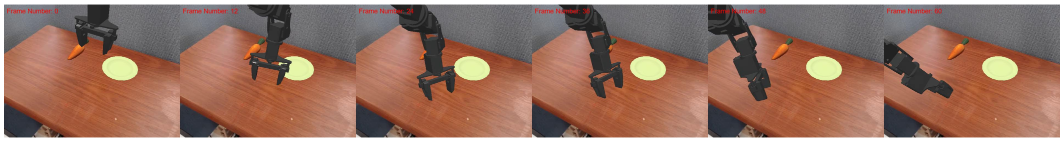


Figure 6: Failure case: The robot arm failed to reach the source object *carrot* using OpenVLA base.

5.3 Dynamics Model

To test the effectiveness of our dynamics model, we evaluate it using Frechet Inception Distance (FID) and Optical Flow Loss (OFL) scores. Frechet Inception Distance (FID) measures how closely predicted future state images resemble ground-truth images, with lower scores indicating more realistic predictions. Optical Flow Loss (OFL) evaluates how well the dynamics model captures temporal changes by comparing pixel-wise motion between consecutive frames, specifically designed to assess whether the model predicts meaningful task-relevant dynamics rather than copying static background information.

Table 3 compares the base model (trained only on the BridgeV2 dataset) and the fine-tuned model (fine-tuned on in-domain SIMPLER trajectories) using the above metrics.

The results in Table 3 highlight both the effectiveness of our base dynamics model and the benefits of fine-tuning. Even without fine-tuning, the Base Model achieves a respectable FID of 17.0, demonstrating that training on large-scale BridgeV2 trajectories allows the model to generalize reasonably well to unseen environments. This baseline performance highlights the generalizability of our model to unseen environments, underscoring the effectiveness of training on diverse, large-scale datasets.

Fine-tuning on in-domain SIMPLER trajectories further improves performance, lowering the FID from 17.0 to 11.2 and reducing the OFL from 1.665 to 0.992. This shows that fine-tuning enhances the model's ability to capture task-relevant temporal dynamics while minimizing redundant static information. These improvements help address the Real2Sim gap by adapting the model to environment-specific dynamics, leading to more accurate and temporally coherent predictions.

Overall, these results demonstrate that our world model provides a solid foundation for dynamics prediction even before fine-tuning, and that the additional fine-tuning step further improves the model's robustness, temporal coherence, and adaptability to the target simulation environment, thereby helping bridge the Real2Sim gap.

6 Qualitative Discussion

We analyze the components of SITCOM with a focus on policy behavior and failure modes.

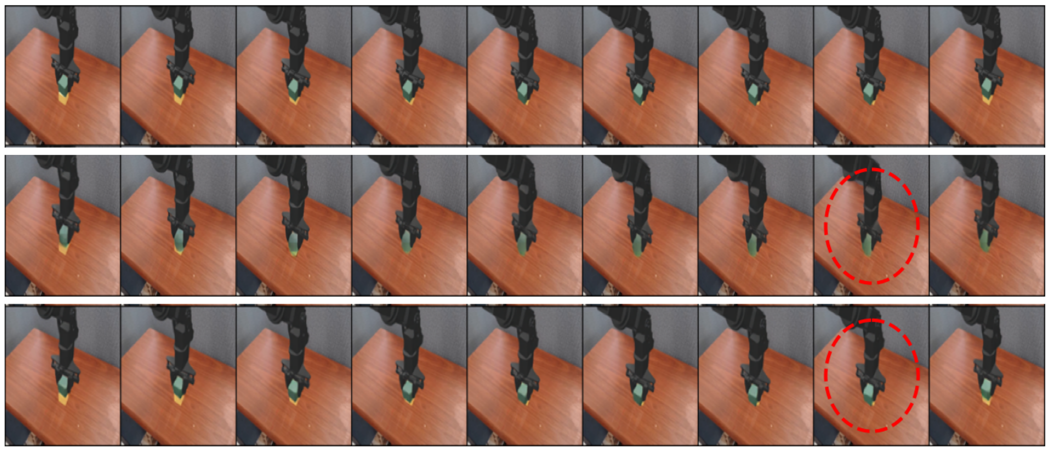


Figure 7: **Qualitative comparison of world model rollouts.** Top row: Ground truth trajectory. Middle row: Rollouts from a world model trained without DAgger-style adaptation. Bottom row: Rollouts from a world model trained with DAgger-style adaptation. Object reconstruction issues without adaptation are highlighted.

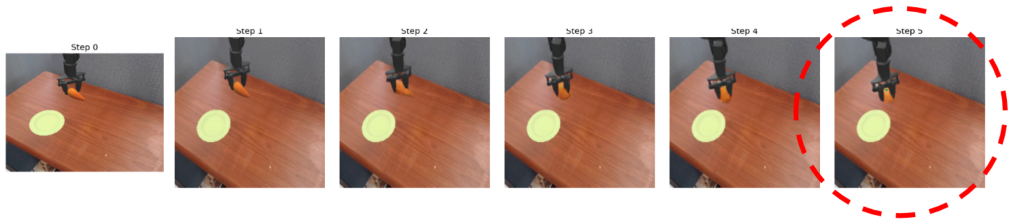


Figure 8: **Example of poor object reconstruction during world model rollout.** Despite improvements from DAgger-style adaptation, occasional failures in object consistency remain evident at longer rollout horizons.

VLA Policy. As shown in Table 2, the baseline OpenVLA [4] achieves a partial success rate of 0.167 but no full completions on the `PutCarrotOnPlate` task, while fine-tuning on SIMPLER trajectories raises the success rate to 0.500, reducing the Real2Sim gap.

Failure Modes. OpenVLA exhibits two main limitations: (i) the **Real2Sim gap**, where differences in dynamics and visual affordances hinder transfer from real data to simulation, and (ii) **limited generalization**, as imitation learning struggles with out-of-distribution scenes. These failures are especially pronounced in fine-grained control tasks such as `PutCarrotOnPlate`, where the agent must grasp at the right affordance and close the gripper precisely; small deviations cause the carrot to slip, as illustrated in Figure 6.

Finetuning OpenVLA to Reduce Real2Sim GAP. We create this model by fine-tuning OpenVLA on ~ 100 trajectories from SIMPLER. The goal is to bridge the Real2Sim gap present in the base OpenVLA. We observe a performance boost of approximately 40% after fine-tuning. The model is now better calibrated for the simulator setting; however, its performance is still around 40%, and we aim for further improvements.

Leveraging Inference-Time Compute Using Simulators for Rollouts. We scale our model using test-time compute through a guidance mechanism that employs an oracle simulator for rollouts. Our reward design incorporates the gripper-object gap, object-destination distance, and grasp success indicators, generalizing well across Bridge dataset tasks. While these rewards may not generalize to all scenarios, they extend to broader robotic manipulation tasks. This approach achieves 80% success on Bridge tasks, significantly outperforming previous methods through effective policy sampling and reward design. Although action selection time increases, the pipeline parallelizes across multiple GPUs to mitigate computational overhead.

World Model. Initially training our world model to predict single timesteps ahead resulted in significant object reconstruction errors during extended rollouts due to compounding prediction errors

(Figure 7, second row). We addressed this using DAgger-style adaptation, where the model uses its own predictions as inputs during training, markedly improving reconstruction over longer horizons (third row). However, prediction drift and reconstruction failures persist during test-time (Figure 8), suggesting policy-generated trajectories remain out-of-distribution for our world model. We attribute this to training exclusively on successful trajectories and hypothesize that incorporating unsuccessful trajectories would mitigate these errors.

7 Future work and Limitations

While SITCOM demonstrates promising results in simulation, several limitations remain, providing opportunities for future research.

1. **Limited Exposure to Failure States**. To address model bias from training solely on successful trajectories, we will augment our dataset with collected failure examples to improve robustness and accurate penalization in off-distribution states.
2. **Real-World Deployment Challenges**. To bridge the sim2real gap, future work will focus on addressing challenges like visual and physical discrepancies through techniques such as real-world fine-tuning, domain adaptation for vision, and closed-loop replanning to correct accumulated errors during execution.
3. **Deterministic Dynamics Model Limitation**. To overcome the limitations of our deterministic dynamics model in handling stochastic environments, we propose exploring probabilistic, action-conditioned video diffusion models. This approach would better capture uncertainty and enable more flexible and robust planning for complex, real-world manipulation tasks by generating diverse and plausible future outcomes.
4. **Inference-Time Bottleneck and Control Frequency**. To overcome the inference-time bottleneck that currently limits real-world control frequency, future work will focus on reducing latency by parallelizing rollout generation and exploring action chunking. Additionally, we propose using action-conditioned video diffusion models to simulate entire trajectories in a single pass, significantly improving computational efficiency for high-frequency, dexterous manipulation tasks.

References

[1] Robert McCarthy, Daniel C. H. Tan, Dominik Schmidt, Fernando Acero, Nathan Herr, Yilun Du, Thomas G. Thuruthel, and Zhibin Li. Towards generalist robot learning from internet video: A survey, 2024. URL https://arxiv.org/abs/2404.19664.

[2] David Ha and Jürgen Schmidhuber. World models, March 2018. URL https://doi.org/10.5281/zenodo.1207631.

[3] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models, 2022. URL https://arxiv.org/abs/2010.02193.

[4] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, and et al. Rt-1: Robotics transformer for real-world control at scale, 2023. URL https://arxiv.org/abs/2212.06817.

[6] Team Octo, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

[7] Manfred Morari and Jay H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4):667–682, 1999. ISSN 0098-1354. doi: https://doi.org/10.1016/S0098-1354(98)00301-9. URL `https://www.sciencedirect.com/science/article/pii/S0098135498003019`.

[8] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale, 2024. URL `https://arxiv.org/abs/2308.12952`.

[9] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[10] Stéphane Ross, Geoffrey Gordon, and Drew 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. JMLR Workshop and Conference Proceedings, 2011.

[11] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pages 906–915. PMLR, 2018.

[12] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[13] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[14] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[15] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[16] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[17] Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

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

[19] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[20] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pages 38–55. Springer, 2024.

[21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[22] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[23] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[24] Daniel Fried, Nicholas Tomlin, Jennifer Hu, Roma Patel, and Aida Nematzadeh. Pragmatics in language grounding: Phenomena, tasks, and modeling approaches. In *Findings of the Conference on Empirical Methods in Natural Language Processing*, 2023. URL `https://arxiv.org/abs/2211.08371`.

[25] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023.

[26] Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023.

[27] Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pages 23301–23320. PMLR, 2023.

[28] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[29] Siddharth Karamcheti, Suraj Nair, Annie S Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang. Language-driven representation learning for robotics. *arXiv preprint arXiv:2302.12766*, 2023.

[30] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, and et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL `https://arxiv.org/abs/2307.15818`.

[31] Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, and et al. Open x-embodiment: Robotic learning datasets and rt-x models, 2024. URL `https://arxiv.org/abs/2310.08864`.

[32] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

[33] Nils Blank, Moritz Reuss, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Wenzel, Oier Mees, and Rudolf Lioutikov. Scaling robot policy learning via zero-shot labeling with foundation models. *arXiv preprint arXiv:2410.17772*, 2024.

[34] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[35] KS Holkar and Laxman M Waghmare. An overview of model predictive control. *International Journal of control and automation*, 3(4):47–63, 2010.

[36] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[37] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[38] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[39] Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B Tenenbaum. Learning to act from actionless videos through dense correspondences. *arXiv preprint arXiv:2310.08576*, 2023.

[40] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

[41] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

[42] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[43] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[44] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

[45] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

[46] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 1 (2):6, 2023.

[47] Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments, 2024. URL `https://arxiv.org/abs/2402.15391`.

[48] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024.

[49] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*, 2023.

[50] Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*, 2023.

[51] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

[52] Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*, 2025.

[53] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024.