Extended Abstract

Motivation Recent works introduce world foundation models (WFMs), which aim to be general multi-modal simulators of the real world by training on massive internet-scale datasets. Building on WFMs, downstream works have proposed world action models (WAMs), which are language-guided visuomotor policies obtained by fine-tuning pretrained WFM checkpoints on small domain-specific expert demonstration datasets. WAMs have been shown to achieve remarkable few-shot generalization performance for manipulation tasks. In practice, however, WAMs can be brittle due to the compounding errors problem, in which distribution shifts during deployment cause a collapse in policy performance. Thus, in the pursuit of robustifying few-shot manipulation, I propose to combat compounding errors via a novel form of online world model-based feedback planning.

Method To overcome the compounding errors problem of WAMs for manipulation tasks, I propose to modify the WAM policy on-the-fly via world model-based feedback planning. Specifically, I propose to plan within a separate, small-scale world model to synthesize corrective behaviors during deployment. Modern WAMs are well-suited to this approach, since they predict future observations alongside action chunks, which can be used as planning targets within a deployment-time planning framework.

At a high level, my approach is as follows. First, I train a WAM on a dataset $\mathcal{D}_{\text{expert}}$ of expert demonstrations for a specific task. Next, I train a small-scale world model on a dataset of diverse demonstrations $\mathcal{D}_{\text{diverse}}$, which can include both expert and non-expert (exploratory) rollouts, so as to cover the relevant task dynamics for downstream planning. Finally, during deployment, I query the WAM for action chunks and predicted observations. Rather than execute these predicted action chunks in open loop, I refine them via planning in the latent space of the small-scale world model, using the WAM's future observation predictions as planning targets. My hypothesis is that this closed-loop planning scheme will improve the robustness of WAM deployment by more effectively constraining the policy to its training distribution during deployment, thus reducing the effect of compounding errors and leading to overall improvements in performance.

Implementation To evaluate my hypothesis, I explore a custom fine-grained block insertion task in the Maniskill simulator. My proposed framework requires three components: the WAM, the small-scale world model, and the planning algorithm. For the WAM, I train a 2B-parameter Cosmos Policy model built on a pretrained Cosmos-Predict2-2B WFM backbone. For the small-scale world model, I train ViT-based predictor and decoder models in the embedding space of the frozen DINOv3 ViT-L model. For the planning algorithm, I use the cross-entropy method (CEM). Since the approach relies on several learned components that inevitably incur errors, I study the effects of these errors via seven experimental conditions that progressively ablate ground-truth oracles with learned ones.

Results By itself, the WAM achieves relatively poor performance, highlighting the issue of compounding errors and leaving a large room for improvement. When the WAM is combined with oracle-based planning, I find that performance is significantly improved, including when the planning seeds and targets are from the WAM. This validates the hypothesis that planning can help reduce compounding errors during deployment and ultimately improve policy performance. Crucially, these gains persist when the oracle model is replaced by a fully learned one, although only when the dataset used to train the world model includes interaction-rich, on-policy data. Qualitative inspections of world model predictions in a decoded pixel space reveal that while large-scale dynamics are well-modeled, fine-grained dynamics, such as the grasp interactions, are difficult to capture if there is insufficient data coverage.

Discussion and Conclusion In this work, I investigate the hypothesis that deployment-time planning can improve the robustness of WAMs by reducing compounding errors. Experimental results strongly support this hypothesis. However, the efficacy of planning is highly contingent on the quality of the world model used for planning. Additionally, CEM substantially increases computational costs during deployment. These limitations motivate several future directions, which include constructing better world model representations (e.g., object-centric ones), considering world model uncertainty during planning (to avoid exploiting learning errors), and gated planning (to avoid unnecessary planning costs when possible).

Improving Fine-Grained Manipulation by World Action Models via Online World Model-Based Feedback Planning

Albert Lin
Department of Aeronautics and Astronautics
Stanford University
albertkl@stanford.edu

Abstract

Recent works introduce world action models (WAMs), which are general-purpose visuomotor policies that are adapted from pretrained world foundation model (WFM) backbones by fine-tuning on small expert demonstration datasets. WAMs have been shown to achieve remarkable generalization at a fraction of standard data costs. In practice, however, WAMs remain brittle due to the compounding errors problem, in which deployment-time distribution shifts cause a collapse in policy performance. In this project, I propose to address this problem via online world model-based feedback planning. My key insight is that modern WAMs predict not only action chunks but also future observations, which can serve as targets for deployment-time planning in an effort keep the policy "on track" and mitigate compounding errors. Concretely, I pair the WAM with a separate, small-scale world model trained on diverse data and refine the WAM's action chunks by planning in the world model's latent space toward the WAM's predicted observations during deployment. I instantiate this framework on a fine-grained block insertion task in ManiSkill, using a 2B-parameter Cosmos Policy WAM, a ViT-based world model operating in a frozen DINOv3 embedding space, and a cross-entropy-method planner. I evaluate the efficacy of the approach across seven experimental conditions that progressively replace ground-truth oracles with learned components. I find that closed-loop planning substantially improves overall policy performance: an oracle dynamics model nearly doubles the policy's success rate, while a learned world model trained on interaction-rich data improves it from 34% to 48%, recovering most of the oracle's gain without any privileged information at test time.

1 Introduction

Recently, world foundation models (WFMs) (Ali et al., 2025; Gao et al., 2026) have been introduced as impressive multi-modal simulators of the real world, obtained by training on massive internet-scale datasets. This has spurred a number of downstream works that construct general-purpose visuomotor policies from pretrained WFM checkpoints using small expert demonstration datasets (Kim et al., 2026; Ye et al., 2026; Pai et al., 2025; Li et al., 2026), which are referred to as world action models (WAMs). WAMs demonstrate remarkable generalization performance in robotic manipulation and have set new benchmark records at a fraction of the standard data costs, leading many to view them as a promising paradigm for the future of household robotics.

However, WAMs can be brittle in practice. Like other policies learned via imitation, they suffer from the compounding errors problem (Yu et al., 2026), where small action errors drive the system into out-of-distribution states and accumulate until performance collapses. An example of this phenomenon is

visualized in Figure 1, where performance collapse of the WAM's observation predictions is evident by clear hallucinations, like duplication of the cube. At the same time, I believe that modern WAMs offer an underutilized resource for addressing the compounding errors problem, as they not only predict action chunks, but also the future observations that they expect to induce.

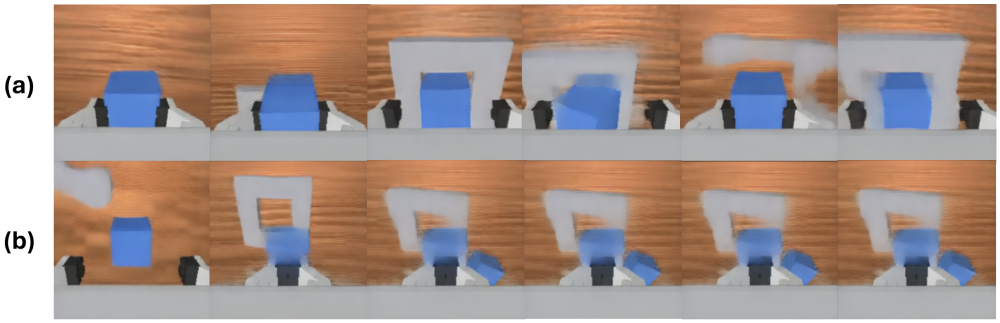

Figure 1: Visualizations of predicted future observations by a WAM in cases where (a) the WAM succeeds, and where (b) the WAM exhibits performance collapse due to the compounding errors problem. Note how, when the model is out of distribution, the WAM's predicted observations exhibit obvious hallucinations like duplication of the cube.

Thus, my key insight is to use these predicted observations to keep the policy "on track" and thereby mitigate compounding errors. Specifically, I propose **world model-based feedback planning**, in which I refine the WAM's predicted actions via planning toward the WAM's predicted future observations within a separate, small-scale world model. Intuitively, since the world model can learn about the dynamics of the system from both expert and non-expert data, this approach leverages off-policy data to bridge the distribution gap.

I instantiate this framework on a fine-grained block insertion task in the Maniskill simulator (Gu et al., 2023), using a 2B-parameter Cosmos Policy WAM (Kim et al., 2026), a ViT-based world model operating in a frozen DINOv3 embedding space (Zhou et al., 2025; Siméoni et al., 2025), and a cross-entropy-method (CEM) planner. Since this approach relies on several learned components that inevitably incur learning errors, I evaluate across seven experimental conditions that progressively replace ground-truth oracles with their learned counterparts, isolating the effects of learning errors on planning efficacy. I find that closed-loop planning substantially reduces compounding errors and improves overall policy performance drastically under oracle planning, even when the planning targets are the WAM's own predicted observations, confirming the key hypothesis. Crucially, these gains are also realizable with a fully learned world model, although not to the same extent, and only if that world model is trained on a dataset that includes interaction-rich, on-policy data. This highlights the importance of accurate dynamics modeling for effective planning.

In summary, my contributions are as follows:

- I introduce **world model-based feedback planning** for WAMs, a deployment-time method that repurposes a WAM's observation predictions as targets for corrective planning within a separate, small-scale world model.
- I instantiate the framework on a fine-grained block-insertion task in Maniskill with a Cosmos Policy WAM, a DINOv3-based ViT world model, and a CEM planner.
- Through a suite of experiments that progressively replace oracles with learned components, I show that planning significantly improves overall task performance, and I identify world model accuracy on the grasp dynamics, and the planner's search efficacy, as the two primary factors that determine success.

2 Related Work

2.1 World Action Models

Recent works establish world action models (WAMs) as a powerful paradigm for visuomotor control, including Cosmos Policy (Kim et al., 2026), DreamZero (Ye et al., 2026), mimic-video (Pai et al.,

2025), LingBot-VA (Li et al., 2026), and others. These models are obtained by fine-tuning pretrained WFM backbones (Ali et al., 2025) on small expert demonstration datasets and demonstrate remarkable few-shot generalization capabilities. WAMs share two important properties relevant for this work: they jointly predict action chunks together with future observations, and during deployment, they execute these action chunks without explicit multi-step feedback planning. Like any other imitation-based approach, WAMs can be brittle during deployment due to the compounding errors problem. To improve deployment-time performance, Cosmos Policy proposes a planning variant that ranks candidate action chunks via a learned value function and executes the highest-scoring chunk. However, training such a value function requires being able to compute task rewards, which is nontrivial in general. In contrast, my work proposes to use an explicit feedback mechanism for planning by using the WAM's own predicted observations as planning targets. I adopt Cosmos Policy as the base WAM in this work, since it achieves state-of-the-art performance at reasonable computational costs.

2.2 Online Planning in World Models

A separate line of work performs online planning in learned latent dynamics models. TD-MPC2 (Hansen et al., 2024) executes MPPI-style trajectory optimization (Williams et al., 2017), guided by a TD-learned value head. SimDist (Levy et al., 2026) extends this to the real world via few-shot real-world adaptation on top of simulation-learned priors. DINO-WM (Zhou et al., 2025) predicts future DINOv2 patch features and plans by optimizing actions to reach a goal image in the embedding space. V-JEPA2 and its action-conditioned variant (Assran et al., 2025) extend this idea to large self-supervised video encoders, again planning to a user-provided goal. LeWorldModel (Maes et al., 2026) demonstrates that small latent dynamics models can be trained end-to-end directly from pixels and used for fast latent planning. However, across all these systems, planning requires either a known scalar reward or a user-supplied goal image. This assumption breaks in the open-set regime that WAMs target, where the task is specified in language, and the relevant goal varies by instruction. To overcome this limitation, I instead propose to use the WAM's own predicted observations as planning targets. I use the DINOv3 ViT-L model (Siméoni et al., 2025) as the encoder for my world model, since it possesses high-quality fine-grained semantic visual understanding.

2.3 Inference-Time Enhancement of Generative Policies

Closest in spirit to this work are recent methods that augment generative robot policies at inference time. CLOVER (a **CLO**sed-loop **V**isuomotor control framework with generative **E**xpectation for **R**obotic manipulation) (Bu et al., 2024) closes the loop on a diffusion-based sub-goal generator by training an inverse-dynamics model (IDM) to follow generated frames, replanning when consecutive sub-goals exceed a distance threshold. Generative Predictive Control (GPC) (Qi et al., 2026) enhances a frozen diffusion policy by either reranking K policy samples through a learned world model or by refining a sampled action via gradient ascent on a learned reward. My framework differs from these works by (i) not requiring a reward function, and (ii) performing explicit planning in a separate world model. This allows my approach to be fully agnostic to task specifications and to most effectively utilize diverse, non-expert data that is often much cheaper to obtain than expert demonstrations.

3 Method

I formulate the manipulation task as a partially observable control problem, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, P)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{O}$ is the observation space, $T(s' \mid s, a)$ is the transition dynamics, and $P(o \mid s)$ is the observation model. I deliberately omit reward modeling, since the deployment-time objective that will be introduced in Section 3.3 relies only on the WAM's predicted terminal observation, not external task rewards. Task progress signals are used only to report evaluation metrics, as described in Section 4, not as part of the planning objective.

As illustrated in Figure 2, the proposed framework consists of three main components: the WAM policy used for nominal control, the world model used for dynamics modeling, and the planning algorithm used for deployment-time action optimization. I describe each of these in detail next.

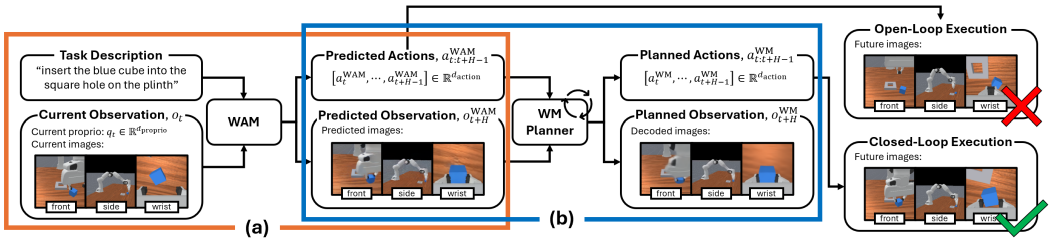

Figure 2: An overview of the framework. (a) A WAM predicts action chunks and the future observations resulting from those action chunks. (b) The action chunks are refined via planning within a world model, where the WAM's predicted future observations serve as planning targets.

3.1 World Action Model (WAM)

The first component of my framework is the WAM, which supplies nominal control. I adopt Cosmos Policy (Kim et al., 2026), a state-of-the-art WAM obtained by fine-tuning the pretrained Cosmos-Predict2-2B video foundation model (Ali et al., 2025) on robot demonstrations. Rather than introduce separate action heads or inverse-dynamics modules, Cosmos Policy encodes the non-visual modalities, e.g., robot proprioception, the action chunk, etc., as additional latent frames injected directly into the video model's latent diffusion sequence, leaving the backbone architecture unchanged. Thus, the model serves as both a policy and an outcome predictor.

I train the model on an expert demonstration dataset $\mathcal{D}_{\text{expert}}$. Concretely, conditioned on the task's language instruction l and on the current observation $o_t = (q_t, I_t)$, where q_t is the robot proprioception, and I_t holds the visual images obtained at system time t, the WAM generates an action chunk $a^{\text{WAM}}_{t:t+H-1} = \left(a^{\text{WAM}}_t, \ldots, a^{\text{WAM}}_{t+H-1}\right) \in \mathbb{R}^{H \times d_{\text{action}}}$ along with the corresponding future observation $o^{\text{WAM}}_{t+H} = I^{\text{WAM}}_{t+H}$, where the horizon H equals the chunk length. Note that since I use only the WAM's predicted future images as the planning target and not the predicted future proprioception, I let $o^{\text{WAM}}_{t+H} = I^{\text{WAM}}_{t+H}$ for simplicity. Also note that o^{WAM}_{t+H} is a single terminal prediction rather than a dense frame trajectory, and that when multiple camera views are available, I^{WAM}_{t+H} comprises the predicted image for each view.

The action chunk $a^{\text{WAM}}_{t:t+H-1}$ defines the nominal behavior, which traditionally is executed without modification and queried at the chunk boundaries. This serves as the baseline against which I evaluate my proposed planning-based approach, in which the predicted future observation o^{WAM}_{t+H} serves as the central corrective signal in my methodology.

3.2 World Model

The second component is the world model, a compact latent dynamics model that, in contrast to the WAM, is trained on a diverse dataset $\mathcal{D}_{\text{diverse}}$ that can include non-expert exploratory rollouts cheaper to obtain than expert demonstrations. The world model architecture follows that in DINO-WM (Zhou et al., 2025), which consists of three main components: a frozen visual encoder ϕ that maps images into a semantic embedding space, a learned latent predictor f_θ that models the dynamics in that latent space, and a learned decoder g_ψ that maps embeddings back to pixels, for human interpretability. All planning in my proposed approach is carried out in the frozen embedding space of the world model.

Encoder. I instantiate ϕ as the frozen DINOv3 ViT-L model (Siméoni et al., 2025), which I choose for its fine-grained semantic features. Specifically, given an image I, the encoder produces a grid of N patch-token embeddings $\phi(I) \in \mathbb{R}^{N \times D}$, where $N = 196$ (a 14×14 grid) at input resolution 224×224, and $D = 1024$. By keeping the encoder frozen, I remove the need to learn a robust visual representation from the limited $\mathcal{D}_{\text{diverse}}$. This also helps embed the WAM's predicted observations $z^{\text{WAM}}_{t+H} = \phi(I^{\text{WAM}}_{t+H})$ in a way that will not degrade based on the limited set of observations in $\mathcal{D}_{\text{diverse}}$.

Predictor. The predictor f_θ is a ViT operating directly on patch tokens produced by the DINOv3 ViT-L encoder. To enforce temporal causality, I instantiate the transformer with block-causal temporal attention. Specifically, within each frame, all patch tokens attend to one another, while across time, a frame attends only to the current and previous frames. To ensure there is sufficient change

observable in each frame transition, I use a fixed frameskip of $\Delta = 5$ (a 4 Hz cadence under the 20 fps recordings). Each frame input to the predictor thus carries the encoded observation (z_t, q_t), comprised of both the visual embedding $z_t = \phi(I_t)$ and the proprioception q_t, as well as the full block of Δ actions executed in that window, $a_{t:\,t+\Delta-1} \in \mathbb{R}^{\Delta\, d_{\text{action}}}$. To combine the different data modalities, q_t and $a_{t:\,t+\Delta-1}$ are each passed through their own learned MLP projector and then concatenated, along the feature dimension, to every visual patch token of the corresponding frame. The predictor outputs the world model's predicted observation Δ frames ahead,

$$o^{\text{WM}}_{t+\Delta} = (z^{\text{WM}}_{t+\Delta},\, q^{\text{WM}}_{t+\Delta}) = f_\theta\Big(\big\{(z_{t-i\Delta},\, q_{t-i\Delta},\, a_{t-i\Delta\,:\,t-i\Delta+\Delta-1})\big\}_{i=0}^{H_{\text{ctx}}-1}\Big), \tag{1}$$

where I index the H_{ctx} context frames by their stride offset $i = 0, 1, \ldots, H_{\text{ctx}} - 1$ from the current frame t, so that frame i holds $(z_{t-i\Delta}, q_{t-i\Delta})$ and its action block $a_{t-i\Delta\,:\,t-i\Delta+\Delta-1}$. Rolling this recursion forward over a candidate action chunk yields the world model's final predicted observation, whose visual component is what the planner in Section 3.3 tries to align with the predicted observation from the WAM. Since the frameskip of 5 does not divide the $H = 16$-step chunk evenly, the predictor advances $\lfloor H/5 \rfloor = 3$ steps, to $t + 5$, $t + 10$, and $t + 15$. Thus, the planner matches z^{WM}_{t+15} against z^{WAM}_{t+16}, a one-frame misalignment that also leaves the final action unoptimized. This inconsistency stems from setting the frameskip to 5 before I noticed the issue. For future work, I intend to correct this by using a frameskip that divides the chunk evenly (e.g., 4, which would yield 4 steps ending exactly at $t + 16$).

Decoder. The decoder g_ψ maps a frame of patch tokens back to pixels, $I^{\text{WM}} = g_\psi\,(z)$, and is trained independently of f_θ via a reconstruction loss. This choice is consistent with DINO-WM's finding that decoding is unnecessary for planning, and that backpropagating the decoder loss into the predictor training can actually degrade planning performance. Since planning operates entirely on embeddings, g_ψ is not used in the planning loop. Instead, I use it solely to render the world model's predicted observations for the qualitative analysis.

Training. I train f_θ on $\mathcal{D}_{\text{diverse}}$ via a multi-step latent prediction loss. Specifically, I slice each trajectory into segments of length $H_{\text{ctx}} + K$, where K is the prediction horizon, and, from the H_{ctx}-frame context, unroll the predictor $K = 2$ steps autoregressively using ground-truth actions. I use a γ-discounted sum of squared errors between the predicted observations and their corresponding frozen-encoder targets, over the K autoregressive steps:

$$\mathcal{L}_{\text{pred}} = \sum_{h=1}^{K} \gamma^{\,h-1} \left(\left\| z^{\text{WM}}_{t+h\Delta} - z_{t+h\Delta} \right\|_2^2 + \left\| q^{\text{WM}}_{t+h\Delta} - q_{t+h\Delta} \right\|_2^2 \right), \qquad z_{t+h\Delta} = \phi(I_{t+h\Delta}). \tag{2}$$

I set the discount $\gamma = 0.95$, which weights the one-step term slightly more than the two-step term. Note that the objective is computed entirely in the embedding space. The decoder is trained separately with the pixel reconstruction loss $\mathcal{L}_{\text{dec}} = \|g_\psi\,(\phi\,(I)) - I\|_2^2$. Full architecture and optimization hyperparameters can be found in Appendix B.

3.3 Planning Algorithm

The final component of my framework refines the WAM's nominal action chunks during deployment by planning within the world model's embedding space. At each action-chunk generation step, using the current observation o_t, I query the WAM once to obtain the nominal action chunk $a^{\text{WAM}}_{t:t+H-1}$ and its corresponding predicted terminal observation o^{WAM}_{t+H}. Then, I encode the latter into the target embedding $z^{\text{WAM}}_{t+H} = \phi(I^{\text{WAM}}_{t+H})$ and search for an action chunk whose world model-predicted outcome best matches that target.

Planning objective. I roll out a candidate chunk $a_{t:t+14}$ through the world model via the autoregressive recursion described in Section 3.2, to obtain the predicted terminal embedding z^{WM}_{t+15}. Then, the planning cost is the mean squared distance between the predicted terminal embedding and the WAM's predicted target in the embedding space, pooled across camera views and patch tokens,

$$C\,(a_{t:t+14}) = \frac{1}{|z|} \left\| z^{\text{WM}}_{t+15} - z^{\text{WAM}}_{t+16} \right\|_2^2, \tag{3}$$

where z^{WM} and z^{WAM} stack the patch embeddings of all camera views, and $|z|$ is their total number of elements. This matches the goal-reaching objective used in DINO-WM but where the goal embedding is supplied online by the WAM's own prediction rather than by an external goal image.

Cross-entropy method. I minimize C using the cross-entropy method (CEM) (de Boer et al., 2005), which is an iterative sampling-based optimizer. At each iteration of CEM, I sample 32 candidate chunks, consisting of the current mean μ together with 31 samples from $\mathcal{N}(\mu, \text{diag}(\sigma^2))$. Since the trailing action is left to the WAM, per Section 3.2, I have that $\mu, \sigma \in \mathbb{R}^{15 \times d_{\text{action}}}$. Out of the 32 candidates sampled at an iteration, I select the 8 lowest-cost elites and refit (μ, σ) to the elite set, and then repeat this process 4 times. To begin the first iteration, I set the initial mean as the WAM's nominal actions, $\mu^{(0)} = a^{\text{WAM}}_{t:t+14}$, and the initial standard deviation as $\sigma^{(0)} = 0.10$ for each action dimension. Although this choice of $\sigma^{(0)}$ effectively explores the continuous end-effector pose delta actions, I note that the gripper command is effectively binary (open or close), for which Gaussian search is a poor fit and thus keeps the gripper close to the WAM's proposal. I defer more effective gripper search for future work. The admissible action range $[-1, +1]$ is enforced by the environment at execution. The returned plan is the final mean $\mu^{(M)}$.

Full-chunk deployment. After the CEM-based optimization, I execute the refined 15 actions $\mu^{(M)}$ followed by the WAM's trailing action before querying the WAM again, matching the WAM's native cadence. Since the world model predictor conditions on a short stride-spaced history of past frames which does not yet exist for the first action chunk, I execute the first chunk without refinement and begin planning only for the second chunk and onward. I find that synthesizing a rest-state history so that the first chunk can also be planned does not improve the overall policy performance and in fact slightly lowers it for the augmented world model condition described in Section 4, so all reported results execute the first chunk without planning. See Appendix A for more details. The no-planning baseline of Section 4 skips the planning step entirely and executes $a^{\text{WAM}}_{t:t+H-1}$ directly.

4 Experimental Setup

Task and environment. I evaluate the proposed framework on a fine-grained cube-insertion task in the ManiSkill simulator (Gu et al., 2023), in which a Panda arm must grasp a cube and insert it into a tight square hole in a fixed mount. The hole is small relative to the cube, so success hinges on precise, contact-rich alignment and insertion. The agent observes RGB images from three 224×224 cameras (a front view, a side view, and a wrist-mounted view) together with a 10-dimensional proprioceptive state (gripper width, end-effector position, and a 6D end-effector orientation). It acts through a 5-dimensional command consisting of deltas in the end-effector position and yaw, plus a gripper open/close signal, which is applied at 20 Hz, with each dimension normalized to $[-1, 1]$. Episodes run for up to 300 control steps (15 seconds), and an episode succeeds if the cube is seated in the hole. To diagnose *where* a rollout breaks down, I also label each unsuccessful episode by the furthest stage it reaches: never lifted the cube, lifted but never aligned over the socket, aligned but never descended into it, or descended but not seated flat.

Models and data. The WAM and the world model use the architectures detailed in Section 3. The WAM is a Cosmos Policy model post-trained on a small expert dataset $\mathcal{D}_{\text{expert}}$ of size 100 (10k frames), collected by a scripted expert that uses ground-truth simulator knowledge. The world model is trained on a diverse dataset $\mathcal{D}_{\text{diverse}}$ that deliberately extends beyond the expert distribution. Specifically, it combines the 100 expert demonstrations with 500 noisy-expert demonstrations (50k frames) and 500 random exploratory rollouts driven by pink action noise (42k frames), for $1,100$ trajectories total. I additionally train an *augmented* world model that adds a fourth source of 500 on-policy rollouts of the WAM itself (134k frames), which are inherently interaction-rich (the policy actively attempts grasps and insertions). Comparing the plain and augmented world models isolates the effect of training on interaction-heavy data.

Experimental conditions. To isolate how each learned component affects planning efficacy, I evaluate the framework across seven conditions that progressively replace ground-truth oracles with their learned counterparts. The swappable components include the **dynamics model** used for planning (the simulator versus the learned world model), the **seed** that initializes the CEM mean (an expert

action chunk versus the WAM's proposal), and the **planning target** (the expert's true future frame versus the WAM's predicted frame). Different selections form different experimental conditions.

Specifically, condition 1 uses the WAM with no planning, which serves as the nominal control baseline. Conditions 2-5 plan with the oracle simulator but progressively move the seed and target from oracles toward learned components: condition 2 uses an expert seed and an expert target, forming a full-oracle upper bound on planning performance; condition 3 uses the expert seed but the WAM's target; condition 4 uses an expert target but the WAM's seed; and condition 5 uses the WAM's seed and target, so that the dynamics model is the only remaining oracle. Conditions 6 and 7 then replace the oracle simulator with the learned plain and augmented world models, respectively, isolating the effect of training on interaction-rich data. The full configuration of each condition is listed alongside its results in Table 1.

Evaluation protocol. Across all conditions, I evaluate on the same 50 task instances for a fair comparison. I use a fixed planner configuration (Section 3.3): 32 samples, 8 elites, 4 iterations, $\sigma^{(0)} = 0.10$, and a cost pooled over all three camera views. I report the success rate as the primary metric, as well as a breakdown over the failure stages defined above.

Computational cost. Training the predictor and decoder each takes roughly 43 hours on a single RTX 4090, while the WAM is post-trained on 8 H100 GPUs for roughly 12 hours. Planning is the dominant runtime cost during deployment. On a single 5090 GPU, the world model-based planner spends roughly 2.4 s per chunk on the CEM search, on top of the WAM's roughly 0.6 s diffusion query, for a total of about 3 s per chunk, which is roughly a quarter of real time relative to the 20 Hz control rate. Oracle-based planning is about $2.6\times$ more expensive, at roughly 6.5 s per chunk, since scoring each candidate requires stepping the simulator and rendering and encoding its terminal frame, rather than rolling out directly in the latent space. Both are tractable for offline evaluation but not yet for real-time deployment.

5 Results

5.1 Quantitative Evaluation

Table 1 reports the success rates and length metrics of successful episodes for each condition, and Figure 3 shows the full distribution of outcomes. The WAM without planning succeeds on only 34% of episodes, confirming that the policy alone is brittle on this contact-rich task. Planning under the oracle simulator (conditions 2-5) raises this substantially, to between 56% and 86%, with condition 2 marking the full-oracle planning ceiling at 86%. Notably, the results suggest that the WAM's own predicted frame is an effective planning target: conditions 4 and 5 differ only in the target source yet perform almost identically (58% vs. 56%), supporting the central idea of planning toward the WAM's predictions. The quality of the seed seems to matter more than the quality of the target (condition 3 achieves a much higher 76%). Since such a seed is unavailable during deployment, condition 5 (56%), which uses the WAM's seed and target, is the ceiling a learned world model should aim for.

Using the learned world model instead of the oracle (condition 6) yields 32%, essentially matching the nominal control, whereas using the augmented world model (condition 7) yields 48%, clearly above the nominal control and recovering much of the way to the oracle ceiling. Figure 3 explains the difference. The task decomposes into two difficult subtasks: the grasp and the insertion, and the conditions diverge in performance mainly on the first one. The plain world model predicts the grasp dynamics so poorly that it fails to lift the cube on 56% of episodes (the large red bar), erasing any benefit from planning. The augmented world model, trained on interaction-rich on-policy rollouts, restores the lift capability to the nominal control's level.

Planning appears to consistently improve the cube insertion performance after the cube alignment is successful. The baseline policy successfully inserts only 61% of aligned cubes, versus 80% and 86% under the plain and augmented world models. An episode-level comparison of the augmented world model against the nominal control confirms that its net gain of seven successes consists primarily of these insertion recoveries. Even oracle dynamics struggle with this insertion. It successfully lifts on 90% of episodes yet still leaves 32% aligned but not inserted properly.

Finally, the distribution of lengths over successful episodes are heavily right-skewed toward the 300-step cutoff, with means clustering near $\sim$140 steps. Under the oracle, planning also leads to *faster* task completion (medians of $77 - 108$ steps versus the nominal control's 124). However, under the learned world models, planning raises the success rate without necessarily improving the speed (medians of $\sim$131, which is comparable to the nominal control). By manual inspection, longer successful episodes exhibit several insertion retries.

Table 1: Success rates and length metrics of successful episodes for each condition, evaluated across 50 episodes. Conditions progressively swap oracle components for learned ones. Condition 2 is a full-oracle upper bound (oracle dynamics, expert seed, expert target); conditions 5-7 use the WAM's seed and target, differing only in the dynamics model used for planning. The length distributions are heavily right-skewed toward the cutoff, so the mean and median diverge.

					Length (success only)	
Condition	Dynamics	Seed	Target	Success Rate	Mean	Median
1	None (open-loop)	—	—	34%	141	124
2	Simulator	Expert	Expert	86%	116	74
3	Simulator	Expert	WAM	76%	102	83
4	Simulator	WAM	Expert	58%	137	77
5	Simulator	WAM	WAM	56%	140	108
6	Learned WM	WAM	WAM	32%	137	132
7	Augmented WM	WAM	WAM	48%	148	131

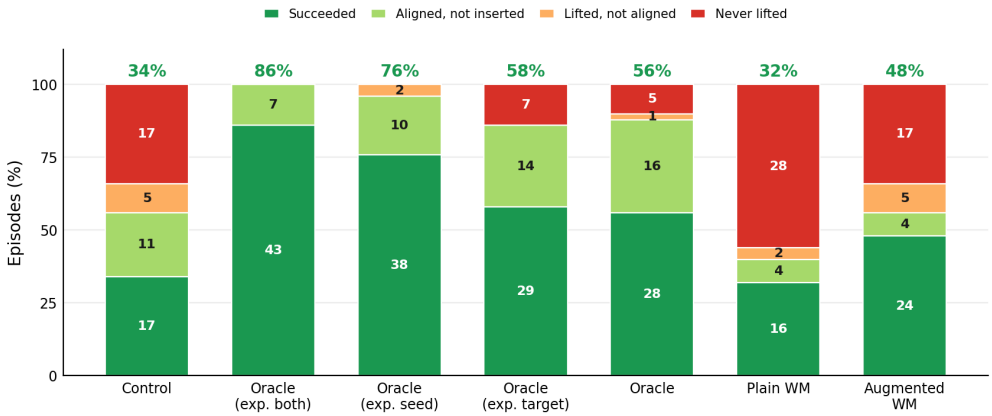

Figure 3: Distribution of episode outcomes across the seven conditions, over 50 episodes, with success rates annotated above each bar. Two subtasks surface as primary failure modes: never lifting the cube (red), and lifting and aligning but failing to insert (light green). The plain learned world model (condition 6) fails most at the lift, the augmented world model (condition 7) handles lifts much better, and the oracle conditions' failures concentrate at the insertion (conditions 2-5).

5.2 Qualitative Analysis

Figure 4 illustrates that the augmented world model is able to more accurately predict the grasp dynamics compared to the plain world model for a held-out WAM evaluation rollout in which the arm fails to lift the cube. Both world models receive the same initial context window and are rolled out autoregressively over the same action sequence, then decoded for inspection. In the ground-truth rollout, the cube stays on the table for the entire episode. The plain world model instead hallucinates a successful grasp, predicting the cube is lifted by the gripper. The augmented world model, trained on interaction-rich on-policy rollouts that include such failed grasps, correctly predicts that the cube remains on the table. This supports the claim that the plain world model's inaccurate modeling of grasp dynamics is what results in the large never-lifted bar in Figure 3. Since the planner scores a candidate by how closely its predicted outcome matches the target, a model that believes every grasp

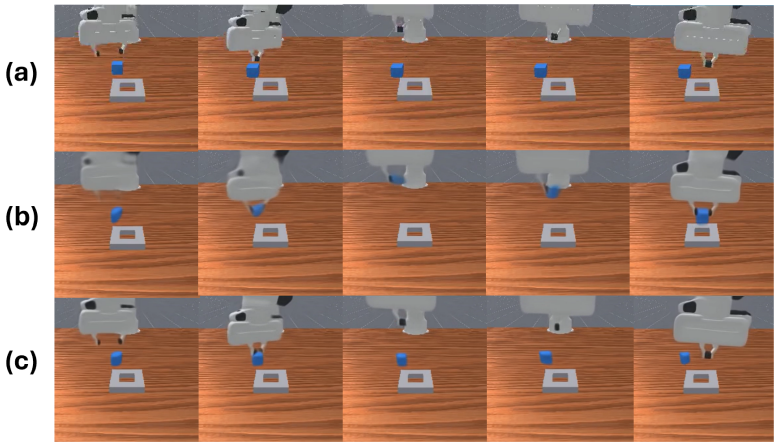

Figure 4: Autoregressive world model rollouts on a held-out WAM evaluation episode in which the arm fails to lift the cube. From an initial context window, each world model rolls out the same action sequence and is decoded to pixels. Columns are rollout steps $h = 0, 3, 6, 9, 12$ (at a frameskip of 5, $h = 12$ is 60 control steps (3 seconds) ahead). **(a)** Ground truth: the policy never lifts the cube, which stays on the table. **(b)** The plain world model hallucinates a successful grasp, predicting the cube is grasped by the gripper. **(c)** The augmented world model correctly predicts that the cube remains on the table. Rows (b) and (c) share the same decoder, so the difference reflects the predictor; the decoder is used only for visualization.

succeeds commits confidently to plans that never materialize, which is what the augmented world model's more accurate predictions would avoid.

6 Discussion

The experimental results strongly support the hypothesis that inference-time planning using a *learned* world model can improve the WAM policy. The augmented world model condition achieves a 48% success rate against the baseline condition's 34%. The deciding factor of performance in this case is the world model's accuracy on the grasp dynamics. The plain and augmented world models differ only in their training data, yet they achieve success rates of 32% and 48%, respectively. Figure 3 localizes the gap largely to the lift subtask, where the plain world model predicts contact poorly enough to fail at lifting the cube in 56% of episodes. Augmenting the training set with interaction-rich, on-policy rollouts restores the lift rate to the policy's own level and is what enables a net performance benefit. Thus, data coverage of the contact dynamics, obtained cheaply here by adding the policy's own rollouts, is an important part of world model-based planning on contact-rich tasks.

The most direct improvement path is likely more interaction-rich data for the world model training. However, suboptimal performance under oracle conditions suggest that there is room for improvement in the planning process. Thus, stronger optimizer, e.g., perhaps model-predictive path integral control, and better world model representations for fine-grained manipulation, e.g., object-centric, are also natural next steps now that a learned model has been shown to support planning at all.

7 Conclusion

In this work, I find that inference-time CEM planning over an augmented DINOv3 world model raises cube-insertion success from 34% to 48% in a contact-rich manipulation task, recovering nearly two-thirds of the gap to a privileged-simulator oracle. The improvement depends on the world model's accuracy on the grasp dynamics, which requires training on the policy's own interaction data. Under even oracle dynamics, the remaining failures seem to be a search problem, which I hypothesize could be due to instability with the planning objective or CEM dynamics. Promising next steps include exploring better world modeling approaches (e.g., object-centric) and improving the planning algorithm itself (e.g., different search strategies or multi-frame objectives).

8 Team Contributions

- **Albert Lin:** Albert is the sole member of this project.

Changes from Proposal Due to computational budget constraints, I decided to drop the comparison against the Cosmos Policy planning baseline, which uses zeroth-order best-of-N sampling-based optimization and thus scales computational costs linearly. I also decided to drop the comparisons of different planning algorithms, since I found that CEM worked well off-the-shelf, so I focused my efforts instead on searching for the best hyperparameters for CEM (see Appendix A) and analyzing the efficacy of the framework under this single planner choice.

9 Acknowledgment of AI Assistance

While all ideas are my own, I would like to be fully transparent about using Claude Opus 4.7 heavily for paper writing and for generating significant amounts of code infrastructure, boilerplate code, data pipelines, and debugging assistance. Thus, I would like to characterize my work clearly as an extensive end-to-end collaboration with AI (and for it to be graded as such). My efforts to ensure that AI did not become a complete solution include: (1) implementing key components along the way (e.g., the paper's losses and algorithms), (2) carefully reviewing and editing all AI-generated content for my full understanding, and (3) pursuing a project scope well beyond what I normally would achieve for a 1-person project, ultimately resulting in a typical level of effort for a course project.

References

Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, et al. 2025. World simulation with video foundation models for physical ai. *arXiv preprint arXiv:2511.00062* (2025).

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. 2025. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985* (2025).

Qingwen Bu, Jia Zeng, Li Chen, Yanchao Yang, Guyue Zhou, Junchi Yan, Ping Luo, Heming Cui, Yi Ma, and Hongyang Li. 2024. Closed-Loop Visuomotor Control with Generative Expectation for Robotic Manipulation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=1ptdkwZbMG

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. 2005. A Tutorial on the Cross-Entropy Method. *Annals of Operations Research* 134, 1 (01 Feb 2005), 19–67. doi:10.1007/s10479-005-5724-z

Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, et al. 2026. DreamDojo: A Generalist Robot World Model from Large-Scale Human Videos. *arXiv preprint arXiv:2602.06949* (2026).

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. 2023. ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills. In *International Conference on Learning Representations*.

Nicklas Hansen, Hao Su, and Xiaolong Wang. 2024. TD-MPC2: Scalable, Robust World Models for Continuous Control. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=Oxh5CstDJU

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. 2026. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163* (2026).

Jacob Levy, Tyler Westenbroek, Kevin Huang, Fernando Palafox, Patrick Yin, Shayegan Omidshafiei, Dong-Ki Kim, Abhishek Gupta, and David Fridovich-Keil. 2026. Simulation distillation: Pretraining world models in simulation for rapid real-world adaptation. *arXiv preprint arXiv:2603.15759* (2026).

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. 2026. Causal World Modeling for Robot Control. *arXiv preprint arXiv:2601.21998* (2026).

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. 2026. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312* (2026).

Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. 2025. mimic-video: Video-action models for generalizable robot control beyond vlas. *arXiv preprint arXiv:2512.15692* (2025).

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. 2026. Inference-Time Enhancement of Generative Robot Policies via Predictive World Modeling. *IEEE Robotics and Automation Letters* 11, 5 (2026), 5534–5541. doi:10.1109/LRA.2026.3673995

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. 2025. Dinov3. *arXiv preprint arXiv:2508.10104* (2025).

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. 2017. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics* 40, 2 (2017), 344–357.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. 2026. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922* (2026).

Anlan Yu, Zaishu Chen, Peili Song, Zhiqing Hong, Haotian Wang, Desheng Zhang, Tian He, Yi Ding, and Daqing Zhang. 2026. WM-DAgger: Enabling Efficient Data Aggregation for Imitation Learning with World Models. *arXiv preprint arXiv:2604.11351* (2026).

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. 2025. DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning. In *Forty-second International Conference on Machine Learning*. https://openreview.net/forum?id=D5RNACOZEI

A Additional Experiments

Initial-chunk planning. The predictor conditions on a short stride-spaced history that does not exist yet at the first action chunk (Section 3.3). To address this issue, I consider two options. The first is to synthesize a rest-state history, where the entire context is duplication of the initial frame, and the synthetic actions hold the arm at its starting pose with the gripper open. The second is to simply skip planning for the initial action chunk, and execute it in open loop instead. Table 2 compares the two on the same 50 episodes for both learned world models. Skipping the initial chunk planning seems to make no difference for the plain world model but helps for the augmented world model. Planning the initial chunk does not change the lift rate (never-lifted is essentially unchanged) but mysteriously degrades the downstream insertion rate. I leave a deeper investigation of this effect to future work. All results in the main text therefore skip planning for the initial chunk.

Table 2: Initial-chunk planning ablation, across 50 episodes, which compares skipping the initial-chunk planning versus planning it from a synthesized rest-state history. All values are percentages of the 50 episodes; the failure columns give the fraction reaching each furthest stage.

World model	Initial chunk	Success	Never lifted	Aligned, not inserted
Plain (cond. 6)	skipped	32%	56%	8%
Plain	planned	30%	56%	8%
Augmented (cond. 7)	skipped	48%	34%	8%
Augmented	planned	30%	32%	28%

Planner configuration. The planner's exploration noise, cost, and camera set were selected via hyperparameter tuning on the simulator oracle (condition 4) and then held fixed across every condition for the results in the main text. After trying $\sigma \in \{0.01, 0.05, 0.10, 0.25\}$, I found $\sigma = 0.10$ to be the best. For the planning objective, I found a spatial patch cost performing at least as well as a class-token cost. Finally, I found that using all three camera views outperformed using only any single view. I therefore use $\sigma = 0.10$, a spatial patch cost, and all three cameras throughout.

B Implementation Details

Table 3 lists the architecture, training, and planning hyperparameters for all learned components. The world model is trained in the frozen DINOv3 embedding space following the DINO-WM recipe (Zhou et al., 2025). The predictor and decoder are trained independently.

Table 3: Hyperparameters for the WAM, world model (encoder, predictor, decoder), and CEM planner.

WAM (Cosmos Policy)	
Backbone	Cosmos-Predict2-2B
Post-training data	100 expert demonstrations (10k frames)
Evaluation checkpoint	iteration 12,000
Action chunk length H	16
Control rate	20 Hz
Action dimension d_{action}	5
World Model (Encoder)	
Encoder	DINOv3 ViT-L/16 (frozen)
Input resolution	224×224
Patch tokens N	196 (14×14 grid)
Embedding dimension D	1024
World Model (Predictor)	
Architecture	ViT: depth 6, 16 heads, head dim 64, MLP dim 2048
Proprio / action embedding dim	64 / 64
Context frames H_{ctx}	3
Frameskip Δ	5
Multistep horizon K	2
Discount γ	0.95
Optimizer	AdamW, lr 5×10^{-5}, weight decay 10^{-3}, grad clip 1.0
Batch / epochs / warmup	24 / 24 / 1 (cosine schedule)
World Model (Decoder)	
Architecture	ViT: depth 4, 8 heads, convolutional output head (dim 1024)
Optimizer	AdamW, lr 3×10^{-4}, weight decay 10^{-5}
Batch / epochs / warmup	16 / 24 / 2 (cosine schedule)
Planner (CEM)	
Candidates / elites / iterations	32 / 8 / 4
Initial std $\sigma^{(0)}$	0.10 (uniform across dimensions)
Actions optimized	15 of 16 (trailing action from the WAM)
Predictor rollout steps per chunk	3
Compute (training)	
WAM	8× H100
Predictor	1× RTX 4090, ∼43 h
Decoder	1× RTX 4090, ∼43 h
Compute (inference)	
Hardware	1× RTX 5090
Planning time, learned world model	∼3 s per chunk
Planning time, oracle	∼6.5 s per chunk