2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

WorldPlanner: Monte Carlo Tree Search and MPC with Action-Conditioned Visual World Models

R. Khorrambakht*[1], Joaquim Ortiz-Haro*[1], Joseph Amigo[1], Omar Mostafa[1], Daniel Dugas[2], Franziska Meier[2], and Ludovic Righetti[1]

***Abstract*— Robots must understand their environment from raw sensory inputs and reason about the consequences of their actions in it to solve complex tasks. Behavior Cloning (BC) leverages task-specific human demonstrations to learn this knowledge as end-to-end policies. However, these policies are difficult to transfer to new tasks, and generating training data is challenging because it requires careful demonstrations and frequent environment resets. In contrast to such policy-based view, in this paper we take a model-based approach where we collect a few hours of unstructured easy-to-collect play data to learn an action-conditioned visual world model, a diffusion-based action sampler, and optionally a reward model. The world model – in combination with the action sampler and a reward model – is then used to optimize long sequences of actions with a Monte Carlo Tree Search (MCTS) planner. The resulting plans are executed on the robot via a zeroth-order Model Predictive Controller (MPC). We show that the action sampler mitigates hallucinations of the world model during planning and validate our approach on 3 real-world robotic tasks with varying levels of planning and modeling complexity. Our experiments support the hypothesis that planning leads to a significant improvement over BC baselines on a standard manipulation test environment.**

I. Introduction

For a robot to operate in real-world environments, it must reason about physical interactions based on raw sensory observations and the actions it takes. A common approach in the literature, known as Behavior Cloning (BC), implicitly encodes this knowledge into a policy by training on offline human teleoperation data [1], [2]. However, such episodic, task-oriented data must meet certain standards (e.g., low entropy [3] and sufficient coverage) and requires frequent environment resets, making it prohibitively expensive to collect at scale. In this paper, we adopt a different formulation: we first learn a dynamics and reward model from unstructured play data and then use it to adapt state-of-the-art planning and control algorithms to synthesize trajectories for new tasks.

Recent advances in diffusion-based generative models [4] have enabled high-fidelity action-conditioned world models [5] with long-horizon predictive capability. Building on this, we propose a framework that synthesizes actions through search in the action/observation distribution of an unstructured play dataset—data that imposes no constraints on entropy or environment resets and can be collected far more easily than task-specific demonstrations.

We show this play distribution contains action primitives sufficient for solving new goal-specific tasks. Specifically, we build a Monte Carlo Tree Search (MCTS) planner and a zeroth-order Model Predictive Controller (MPC) around a compact diffusion-based visual world model [5] and image-conditioned reward functions, trained from a few hours of play data on a single GPU. While we use a small task-specific model to validate our dataset-driven hypothesis, the framework applies directly to larger foundation world models. We validate on real-world manipulation tasks spanning rigid and deformable objects. Our contributions are as follows:

- We propose a control and planning framework based on learned world models from *unstructured play data*, avoiding the costly requirements of collecting goal-directed optimal demonstrations and enabling the synthesis of behaviors not directly seen in the play data.
- We propose to discretize the MCTS search space using a stochastic diffusion model capturing the play distribution. This minimizes out-of-distribution rollouts by constraining search in the training distribution of the world model, thus enabling reliable planning.
- We conduct our study using real hardware/data and demonstrate quantitative/qualitative validations in manipulation tasks with varying levels of complexity.

Note that our framework is complementary to model-based RL methods such as Dreamer [6] and TD-MPC2 [7], which jointly learn task-specific policies through online RL. WorldPlanner instead operates at inference time using only play data, with task specification provided entirely through a reward model or goal image.

II. Related Work

A. Short Horizon Planning with World Models

To our knowledge, most contributions focusing on the runtime generation of actions with world models are limited to short-horizon planning, either due to error accumulation in traditional world models with highly compressed latent spaces [6], [8], [9] or due to the prohibitive compute requirements of more recent foundation world models [10]. A recent representative example is [11], which models the dynamics in the latent space of a foundation video encoder and uses it to formulate a one-step MPC planner. The authors of [11] also report one-step MPC with Cosmos foundation world model [10], in which case each control update takes up

[1]Center for Robotics and Embodied Intelligence, Tandon School of Engineering, New York University, Brooklyn, NY
[2] FAIR at Meta. Franziska Meier and Daniel Dugas contributed in an advisory capacity.
Code is available at `https://github.com/machines-in-motion/WorldPlanner`.

to several minutes, demonstrating the need for more efficient models in planning. In this paper, we show that a relatively small diffusion-based world model originally proposed for simple Atari games [12] can instead be used to learn high-fidelity world models for real-world robotic tasks, avoiding the prohibitive computation cost due to its size and ensuring long-horizon simulation due to the diffusion retraction to the image manifold at each prediction step. Additionally, we train our world model from scratch from a small-scale (few hours) dataset, showing the possibility of adoption as a local visual dynamical model for planning and control.

B. Long Horizon Planning

The prior research on real-world robotic long-horizon planning with visual world models is highly sparse, with only very recent examples such as [13] that adopt a simulator as the world model and a pretrained vision-language-action model as an action proposal module for the MCTS. Alternative examples include the AlphaGo line of work [14] that proposes the adoption of learning in MCTS in achieving remarkable play performance in game environments. To our knowledge, this paper is the first showing long-horizon multi-step planning in real-world robotic settings using learned world models. It demonstrates that with suitable world and action proposal models, such run-time reasoning and planning solutions may also be adopted to solve robotic tasks. Furthermore, we demonstrate that planning in low-level action space may be rendered feasible by discretizing the action space using a diffusion action model trained on unstructured play datasets.

C. Model-Based RL with Latent Dynamics

A related family of methods—including Dreamer [15], DreamerV3 [6], and TD-MPC/TD-MPC2 [7], [8]—uses learned world models as simulators within an online reinforcement learning loop. These methods jointly train a latent dynamics model, a value function, and a policy through TD learning or latent imagination, demonstrating strong sample efficiency on continuous control benchmarks. TD-MPC2 in particular performs local trajectory optimization via CEM in the latent space of a decoder-free world model augmented by a terminal value function, and has been shown to scale across a wide range of tasks and embodiments [7].

These methods differ from ours in three key ways: (1) they require per-task online RL training, while our framework uses only reward-free play data; (2) TD-MPC2's decoder-free implicit latent model precludes the visual rollouts and image-manifold retraction needed to suppress hallucination over long horizons; (3) they perform short-horizon local planning, whereas we use global MCTS tree search guided by the action prior. We therefore view them as complementary—joint task learning with model assistance—rather than direct alternatives to our inference-time, play-data-driven planning framework.

III. Method

Our method consists of two stages. In the first stage, we train a world model, a reward model, and an action prior model from a dataset of unstructured play trajectories. In the second stage, we use these models to implement an MCTS planner and an MPC controller to generate new motions. We present the notation and formal problem definition in Sec. III-A, describe the learned models in Secs. III-B, III-C, and III-D, and then introduce the MCTS planner in Sec. III-E and the MPC controller in Sec. III-F.

A. Notation and Problem Definition

We use standard MDP notation $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho, \gamma)$, where $s \in \mathcal{S}$, $a \in \mathcal{A}$, $\mathcal{P}(s' \mid s, a)$ is the transition density, $\rho(s)$ the initial-state distribution, and $r(s, a)$ the reward. The goal is to find a policy π (global planner + local controller) maximizing:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{H} \gamma^{t} r(s_t, a_t)\right].$$

We learn $\mathcal{P}$ from a dataset of *non-goal-conditioned* play trajectories:

$$\mathcal{D} = \{\tau_1, \ldots, \tau_M\}, \quad \tau_i = \{(s_0, a_0), \ldots, (s_N, a_N)\},$$

where s are scene images and a are end-effector velocity commands. Our objective is to generate plans transitioning from a start state s_s to a goal s_g.

B. Diffusion World Model

We adopt the formulation proposed by the task-specific DIAMOND Atari world model [5] due to its high visual fidelity and its fast training and inference on a local GPU. Furthermore, because DIAMOND is a diffusion generative model, it can capture the multi-modality of play behavior.

The goal of this world model is to capture the distribution of future states s_{t+1} conditioned on a history of past actions and states $(s_{t:t-h}, a_{t:t-h})$. Specifically, we train a denoiser D_θ parametrized by θ and trained by optimizing the following score-matching loss [5]:

$$\mathcal{L}(\theta) = \left\| D_\theta(s^{\tau}_{t+1} | s_{t:t-h},\ a_{t:t-h}) - s_{t+1} \right\| \tag{1}$$

where s^{τ}_{t+1} is the noisy version of the clean s_{t+1} at diffusion step τ. During inference, we sample the next state by iteratively solving the reverse diffusion process. It is important to note that planning and control require the quick generation of many parallel rollouts; thus, denoiser evaluation in sampling the next states has to be kept as small as possible. The noise scheduling, network preconditioning, and integration paradigm adopted by DIAMOND is based on EDM [4] and enables high-quality sampling only with a few (3) denoising steps.

The denoiser is realized using a U-Net [16] conditioned on the image history stacked channel-wise and concatenated with the noisy image input, while action and diffusion step conditioning is realized through adaptive group normalization [17]. If the setup is comprised of more than one camera views, the RGB images are stacked channel-wise and treated as a single image with $n_{view} \times 3$ channels.

Our formulation is aligned with recent foundation world models for robotics [10], [18]; we deliberately use a small

task-specific model to evaluate our dataset-driven hypothesis under constrained compute, while noting that such a model could also serve as a local surrogate distilled from a larger one.

C. Action Generator Model

The distribution of robot actions in the unstructured play dataset $\mathcal{D}$ contains meaningful interactions and motion primitives that can later be combined through planning to solve new tasks. Thus, in this project, we also learn a play action policy, denoted as $\pi_{prior}(a_t \mid s_t)$, using the same play dataset $\mathcal{D}$. Specifically, we use a diffusion policy [1] to achieve diversity and multimodality. Together with the world model, the stochastic action policy enables sampling of short trajectories of action/state pairs $a_{t:t+H}, s_{t:t+H}$, which will be used in Sec. III-E to expand the search tree. An additional benefit of learning the action distribution is that the sampled actions remain within the training distribution of the world model, thereby minimizing the probability of hallucination.

D. Reward Models

Evaluating the rollouts generated during planning and control in the world model will require an image-conditioned reward function. In this paper, we consider three approaches to formulate image-space reward functions:

1) Fully Geometric: For tasks where the objective is explicitly representable geometrically, an off-the-shelf object pose tracker (position and orientation) can be adopted as part of the reward function. Specifically, the tracker $f_\psi(s_t)$ is incorporated as input to an explicit reward function $r(f_\psi(s_t), a_t)$ that formulates a geometric objective of the task (e.g. position difference).

2) Latent Space Image Distance with Pretrained Visual Foundation Models: A more general approach represents the goal as an image s_{goal} and encodes both the current and goal images into a high-dimensional embedding space, where the geometric distance (e.g., cosine similarity or Euclidean) reflects the semantic or geometric difference between the two images, defined as:

$$r(s_t, a_t | s_{goal}) = r_a(a_t) - \alpha \left\| f_\psi(s_t) - f_\psi(s_{goal}) \right\| \quad (2)$$

where $r_a(\cdot)$ is an action-dependent reward term (e.g., the squared norm of the action), and α is a tunable hyperparameter. In our experiments, $f_\psi(s_t) : \mathbb{R}^{W \times H \times C} \to \mathbb{R}^d$ is a pretrained foundation image encoder (DINOv2 [19]). We additionally propose adopting an optional object-centric attention mechanism that retains only the object(s) of interest in the image before computing the $f_\psi(\cdot)$ embedding. In this case, the DINOv2 distance is ensured to encode only the dissimilarity of the objects of interest between the query and goal images. We empirically observed that this filtering improves planning performance and convergence rate.

3) Training Reward Functions from Video: A reward model encoding temporal progress can be trained on a passive dataset of observations of the robot interacting with the environment during play or while performing specific tasks. Specifically, given a start and goal state s_{start}, s_{goal} sampled from temporal sequences of states, the objective is to learn an energy function $f_\psi(s_t \mid s_{start}, s_{goal})$ that assigns higher values to states s_t that are temporally closer to s_{goal}. We adopt the Bradley–Terry $\mathcal{L}(\psi)$ objective from [20] to train this reward model:

$$\begin{aligned} \mathcal{L}(\psi) = & \frac{e^{f_\psi(s_i)}}{e^{f_\psi(s_i)} + e^{f_\psi(s_j)}} \mathbf{1}\{i > j\} \\ & + \frac{e^{f_\psi(s_j)}}{e^{f_\psi(s_i)} + e^{f_\psi(s_j)}} \mathbf{1}\{i \leq j\}, \end{aligned} \quad (3)$$

where s_i and s_j are states randomly sampled from a video chunk whose first and last frames are s_{start} and s_{goal}. The learned energy function f_ψ takes DINOv2 features as input instead of raw images. For clarity, s_{start} and s_{goal} are omitted from the notation of the energy function $f_\psi(\cdot \mid s_{start}, s_{goal})$ in Eq. (3).

Algorithm 1: Global Planner: (MCTS)

Input: Initial state s_0, prior policy π_{prior}, world model $\mathcal{P}$, reward model r, horizon H_{sim}, rollout number M_{sim}, exploration constant c, minimum visit count for solution node n_{min}

Output: Plan from root to node with maximum average value

Initialize root node $v_0 \leftarrow s_0$ with $n_{visit}(v_0) = 0$, $V_{total}(v_0) = 0$

Expand v_0 using π_{prior} and $\mathcal{P}$

while *termination criterion not reached* **do**

- `// Selection`
- Traverse from v_0 to a leaf by choosing child with maximum UCB1 (Eq. (4))
- `// Expansion`
- **if** v *visited* **then**
  - Expand via $\pi_{prior}, \mathcal{P}$; set v to first new child
- `// Simulation`
- From v, run M_{sim} rollouts to horizon H_{sim} using π_{prior} and $\mathcal{P}$; compute R via Eq. (5)
- `// Backpropagation`
- For each node u on path $v \to v_0$: update $V_{total}(u) \leftarrow V_{total}(u) + R$, $n_{visit}(u) \leftarrow n_{visit}(u) + 1$

return Plan from root to node with maximum average value and $n_{visit} > n_{min}$.

E. Global MCTS Planner

In this section, we formulate the MCTS algorithm using the learned world model as a simulator. The goal of the planner is to take the robot from its initial state s_s to a goal state s_g. Starting from the initial state, MCTS builds a search tree where the nodes of the tree are states, and the edges are short trajectories generated by rolling out the prior policy π_{prior}.

The tree is built using the classical MCTS steps: traversal, expansion, simulation, and backpropagation.

1) In the *traversal step*, the search begins at the root and identifies the most promising node to evaluate or expand. The notion of "best" is quantified through the Upper Confidence Bound (UCB1), calculated as follows for each node:

$$UCB1(node) = \frac{V_{total}}{n_{visit}} + c\sqrt{\frac{\log(N)}{n_{visit}}} \tag{4}$$

where n_{visit} is the number of visits to the node, N is the total number of visits to the parent node, c is a constant determining the exploration–exploitation balance, and V_{total} is the estimate of the value function.

2) In the *expansion step*, π_{prior} generates short action sequences from a leaf node, rolled out in parallel in the world model; the final states become new tree nodes, and the branching factor is a user-defined parameter.

3) The *simulation step* runs M_{sim} parallel rollouts of H_{sim} steps from a newly added node. The score exploits the joint stochasticity of the world model and action prior to explore diverse action modalities:

$$\max_{\substack{T\in\{0\cdots H_{sim}\}\\ k\in\{0\cdots M_{sim}\}}} \sum_{t}^{T} r(s_t^k, a_t^k). \tag{5}$$

4) In the *backpropagation step*, the score propagates up the path to the root, updating V_{total} and n_{visit} for each ancestor. These steps repeat until the maximum number of iterations is reached or a node with sufficient value and $n_{visit} > n_{min}$ is found (Alg. 1).

F. Local MPC Controller

Open-loop execution of the plan from the previous section may deviate from the desired motion due to cumulative noise and model errors. Addressing this problem requires closing the visual feedback loop, either by distilling the planner into a neural policy or by taking the plan as input to an image-based MPC formulated using the same world and value models. In this paper, we focus on the latter solution as it enables direct online movement generation.

We use a zeroth-order optimizer [21] to leverage cost smoothing, improved performance in avoiding local minima compared to gradient-based counterparts (especially for neural components with noisy gradients), and to avoid the need for computing gradients from the world and reward models. Further, this approach can easily be parallelized on GPUs.

Starting at $t = 0$, and a plan from MCTS (i.e., a sequence of actions and states) $\tau^{MCTS} = \{(a_0^P, s_0^P), \cdots (a_{h_P}^P, s_{h_P}^P)\}$ with length H_P we define the procedure as follows:

1) Population Generation: For each timestep in the window $\{t, \cdots, t + H_{MPC}\}$, we define a Gaussian $\mathcal{N}(\mu_t, \Sigma_t)$ centered at the planned action $\mu_t = a_t^P$, with diagonal $\Sigma_t \in \mathbb{R}^{n_u \times n_u}$. We draw K trajectory samples $\mathcal{A}^{plan} = \{A^k\}_{k=1}^K$, augmented with K' zero-mean trajectories $\mathcal{A}^0$ (fixed Σ_0) to encourage exploration.

2) Rollout and Evaluation: Each sample from $\mathcal{A}^{plan} \cup \mathcal{A}^0$ is rolled out in the world model from the current observation s_{cam}, yielding rewards $\mathcal{R} = \{R^k\}$ that combine an action cost and a plan-following cost.

Algorithm 2: Local Zeroth-Order MPC Controller

Input: MCTS plan τ^{MCTS}, window H_{MPC}, execution chunk h_{exec}, populations K, K', elites K_{elite}, iterations N_{MPC}, distance metric d, max deviation d_{max}
Output: Executed action sequence with visual feedback corrections

```
Initialize time index t ← 0
while t < H_P do
    // Population Generation
    Around planned actions in window {t, ..., t + H_MPC}, define Gaussians N(μ_t, Σ_t) with μ_t = a_t^P
    Sample K trajectories A^1, ..., A^K and add K' zero-mean trajectories
    // Iterations
    for i ← 1 to N_MPC do
        // Rollout and Evaluation
        For each A^k, rollout with π_prior and P from s_cam; compute reward R^k
        // Update
        Select top-K_elite samples by R^k
        Update μ_t, Σ_t via weighted mean/variance (Eq. (6)), enforce diagonal Σ_t and minimum variance
        // Resample
        Sample K + K' action trajectories using the updated distribution parameters.
    // Execution
    Execute first h_exec actions on robot; capture new s_cam; Update t.
    // Termination Check
    if d(s_cam, s_t^P) > d_max then
        Trigger MCTS re-planning; break
```

3) Update: We adopt the CEM update rule [8]: the top-K_{elite} samples by reward update μ_t and Σ_t as weighted sample means and variances:

$$\begin{aligned} \mu_t &\leftarrow \frac{\sum_{m\in\ elites} R^m a_t^m}{\sum_{m\in\ elites} R^m}, \\ \Sigma_t &\leftarrow \frac{\sum_{m\in\ elites} R^m (a_t^m - \mu_t)(a_t^m - \mu_t)^\top}{\sum_{m\in\ elites} R^m} \end{aligned} \tag{6}$$

For simplicity, we force the off-diagonal elements of the covariance matrix to zero, and to avoid exploration collapsing to zero, we enforce a minimum diagonal covariance at each step. Note that if we set $K_{elite} = 1$ and Σ_t to a fixed baseline Σ_0, we recover the predictive sampling method [22]. Also note that optimizing directly over actions may produce non-smooth results, so in practice we optimize over the knot points of a smooth curve [22].

After N_{MPC} iterations, the first h_{exec} actions are executed, a new observation s_{cam} is captured, and the loop continues

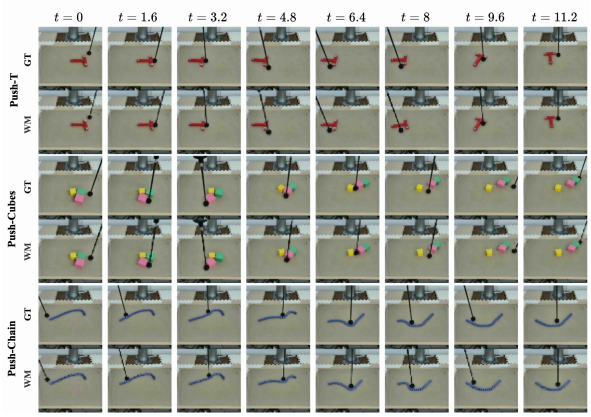

Fig. 1. World model rollouts vs ground truth for all the tasks studied.

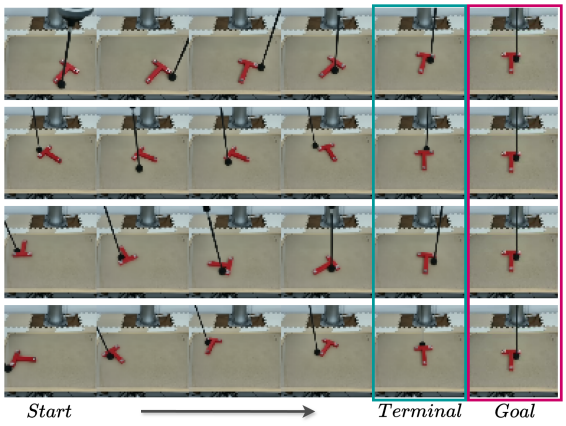

Fig. 2. Examples of the MCTS plans for the push-T task aiming to push the object to the center of the board.

until the goal is reached or deviation exceeds d_{max}, triggering MCTS re-planning.

IV. Results

A. Setup and Training Details

All our models are trained on real-world data collected with a 7-DoF robotic arm (Flexiv Rizon-10S). Specifically, the robot is teleoperated by a user to perform unstructured and randomized interactions with the scene for roughly four hours. Note that this dataset is allowed to have high entropy (which is not desirable in BC datasets [3]) and does not require environment resets, making it much easier to collect compared to task-specific behavior cloning datasets. Using this data, we train the world model, the prior play policy π_{prior}, and the reward model described in Sec. III.

The world model is trained on a single NVIDIA RTX 4090 GPU with 24 GB of RAM and requires roughly three days to achieve acceptable long-horizon rollouts. For the action prior, we use the LeRobot [23] implementation of the diffusion policy and observe stable random exploration behavior after less than one hour of training. Finally, the reward function $f_\psi(.|s_{start}, s_{goal})$ described in Sec. III-D.3 is implemented as a ViT [24] with learned positional embeddings. It takes as input the batch embeddings of the start, goal, and query images, and predicts the corresponding value as output. This encoder is trained by maximizing Eq. (3) on random states drawn from interaction video sequences of length 12.8 s, with convergence observed after roughly 12 hours. Additionally, the geometric object tracker used in Sec. III-D.1 is a CNN trained on a small independent robot-object interaction dataset with labels provided from an AprilTag-based pose tracker.

B. Environments and Corresponding World Models

We evaluate our method for three categories of tasks with evolving degrees of modeling and planning complexity. The *push-T* task, being the simplest, requires capturing the interaction between the robot and a single rigid-body object sliding on a surface. The next level is the *push-cubes* tasks that additionally includes inter-object collisions. Finally, the *push-chain* task considers manipulating deformable objects. Together, these tasks demonstrate that with the same world model formulation, a wide variety of complex real-world dynamic effects can be modeled. Fig. 1 shows the trained world model rollouts for each of these environments. The comparison between the Ground Truth (GT) and World Model (WM) rollouts in Fig. 1 demonstrates that the predictions remain highly consistent with the state of the world even after 11.28 seconds (64 steps) of auto-regressive forward integration.

C. Single-Object/Robot Interaction

We consider the task of pushing a T-shaped tool from a random initial pose to the center of the board as a simple controlled environment. This task represents the complexities of contact modeling and sequential decision making while being structured and simple enough to evaluate quantitatively using geometrical metrics (Sec.III-D.1). Qualitative examples of the generated plans are demonstrated in Fig. 2 and show the planner's ability in determining sequences of contact points and motions that solve the task.

Next, we conduct a quantitative study to compare the MCTS planner against BC baselines on 100 random initializations of the T-tool pose. To ensure identical initialization for all baselines, we use the world model as a simulator and sample 100 random states of the board from the play dataset as start states. We also compare our MCTS planner with two types of rewards against two BC policies based on transformers (ACT [2]) and diffusion modeling (diffusion policy [1]). The BC baselines are trained on fewer than 100 high-quality human demonstrations independent from the play dataset used to train the world model. Each of the two MCTS baselines respectively employs the geometrical reward in Sec. III-D.1, and the video reward model in Sec. III-D.3. Notably, the latter is trained on the same demonstration dataset used to train the BC baselines (only passive video observations used to train the value model) to maintain fairness in comparisons against the BC baselines.

For each random starting configuration, we evaluate the BC policies five times for a duration equal to the longest plan generated by MCTS. For both the planner and the BC

TABLE I

SUCCESS RATES (%) OF MCTS COMPARED TO BC BASELINES ON 100 RANDOM SETTINGS OF THE PUSH-T TASK.

Method	≤ 2.5 cm	≤ 5 cm	≤ 7.5 cm	≤ 10 cm
MCTS+Video Reward	69%	92%	95%	97%
MCTS+Geometric Reward	52%	81%	86%	91%
Diffusion Policy* [1]	49%	70%	83%	88%
ACT* [2]	32%	54%	57%	60%

*Diffusion Policy and ACT are trained on goal-directed data, while the world model used with MCTS is trained on play data.

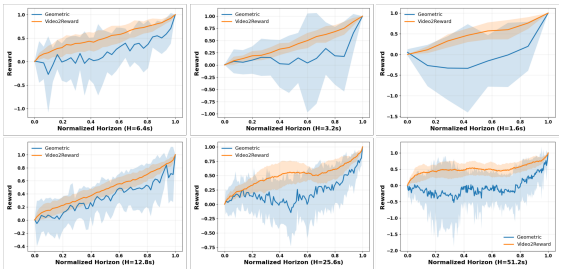

Fig. 3. The evolution of the normalized reward predicted by the geometrical reward model (Sec. III-D.1) and the reward learned from passive videos (Sec. III-D.3) on 10 random selections of 6 temporal lengths.

baselines, a trial is considered successful if, at any point during the plan, there exists a state with a yaw rotation error smaller than 0.3 radians and translation errors smaller than $e_d \in \{2.5, 5, 7.5, 10\}$ centimeters. The results are shown in Table I, where MCTS achieves a higher success rate across all threshold levels. We hypothesize that this improvement is due to the planner's ability to generate new behaviors online, even if they were not demonstrated in the training data. In contrast, a BC policy simply memorizes demonstrations and, when encountering mistakes, cannot inherently recover if corrective behaviors were absent during training.

We note the comparison is asymmetric in data *nature*, not volume. Play data is goal-independent and reset-free, making it far cheaper to collect than task-specific BC demonstrations, and would be trivially available alongside any BC dataset. The comparison illustrates what is achievable *without* high-quality demonstrations.

Running the MCTS using the video to reward model in Sec. III-D.3 leads to the highest performance. Fig. 3 shows the value distribution predicted by the learned reward model on sequences of different temporal lengths (10 random runs per sequence length). As can be seen, the learned reward model exhibits clear, low-variance monotonic progress as the query state approaches the goal, while the geometric metric only considers the geodesic distance to the goal and does not assign value to the intermediate transition states required to reach the goal state. Note that, as expected, when the trial sequence lengths exceed the video lengths (12.8 s) used to train the model, the predictions deviate from a monotonic linear trend but still remain approximately monotonic.

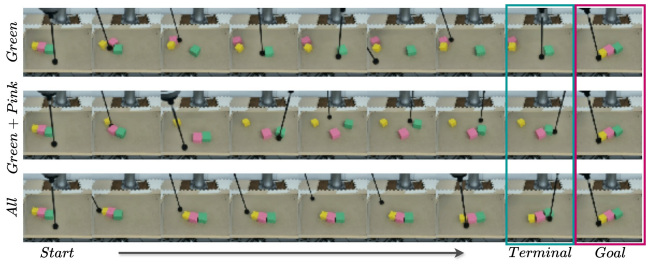

Fig. 4. The plans generated with the world model and MCTS. The inter-object contacts are leveraged to move the cubes, while the masked DINOv2 reward model enables generating plans for selective manipulation of objects of interest in the scene.

D. Multi-Object Selective Interaction

One advantage of training a world model and using planning is the ability to optimize for diverse objectives. Here, we show that by changing the reward, we can generate plans to solve different tasks: either moving three cubes to the target position or moving a single cube. To modulate this objective, we use the masked DINOv2-based reward model presented in Sec. III-D.2, where the object of interest is selected in the image using the SAM2 model [25], and the reward is computed from the embedding distances between the masked states. With this reward, the planner is capable of manipulating the object of interest until it matches its corresponding configuration in the goal image. Importantly, in the generated plans, other objects may also be used if they help indirectly push the object of interest. Fig. 4 shows a qualitative visualization of plans for this task.

E. Deformable Objects Interaction

We now use our planner to manipulate a flexible chain by pushing it until it adopts a desired form, as shown in a goal image (Fig. 1, *push-chain* environment). We employ the same masked DINOv2-based reward as in the previous section, masking out everything except the chain before feeding the DINOv2 encoder for distance computation. As shown in Fig. 6, two independent runs yield different strategies: B_1 pushes the chain right then nudges from the opposite side, while B_2 pushes right and corrects via vertical motions. This stochasticity stems from both the action prior's high-entropy sampling and the diffusion world model's multi-modal distribution, where small perturbations during rollout can trigger different modes.

1) Problem Length vs Performance: Random chunks from the play dataset define problems of varying difficulty (longer chunks considered more difficult), with first and last frames serving as start and goal states. We compute two metrics: IoU measures overlap of the terminal-state chain with the goal image, and is sensitive to both planning performance and chain geometry preservation (e.g., length/width variations due to hallucination); coverage reports the fraction of the goal-state chain overlapped, and is less sensitive to hallucination-induced deformations. Fig. 7 shows the expected negative correlation with problem length, while

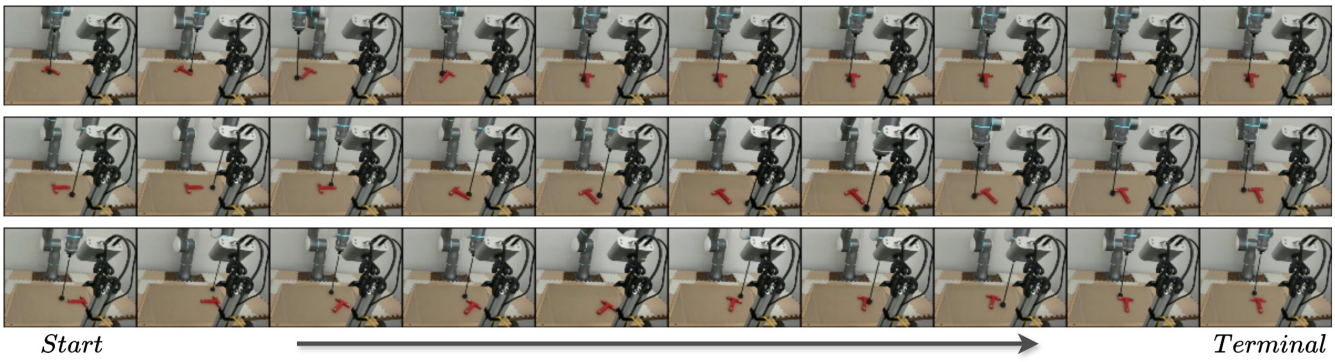

Fig. 5. MCTS plans executed on the real robot using the closed-loop image-space MPC. Each row shows the snapshots of a plan with a different initial object pose.

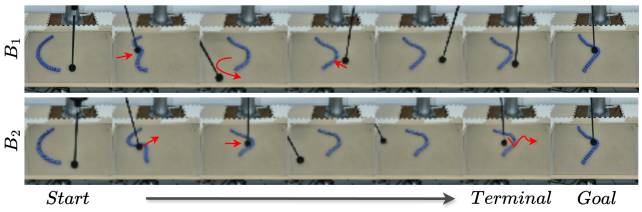

Fig. 6. Two successful trajectories generated by the MCTS planner with identical start and goal states. Due to the stochasticity of the sampler and multi-modality of the diffusion world model, multiple runs can lead to multiple valid solutions (B_1, B_2). Red arrows are added manually to make the end effector movement easier to interpret.

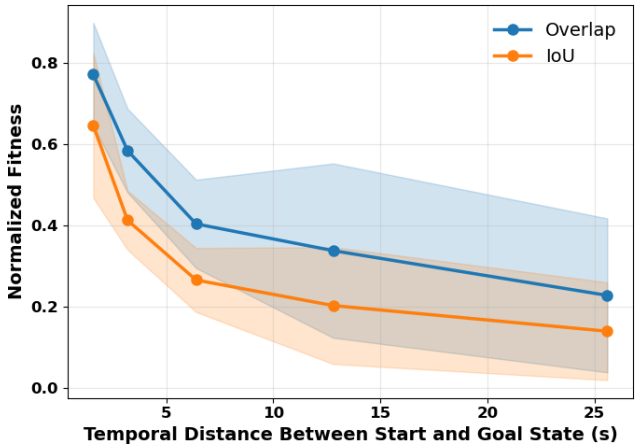

Fig. 7. The similarity between the rope in the terminal states of the plan and the goal image for different problem lengths. The IoU is more sensitive to object deformation during planning.

Fig. 8 confirms that even worst-case terminal states remain qualitatively close to the goal.

F. Closed-Loop Plan Execution with MPC

MCTS plans are tracked by the MPC controller (Sec. III-F), which absorbs small disturbances without requiring re-planning at every step. We demonstrate this on the *push-T* task, where plans are generated in under five minutes on average (80–300 MCTS rollout steps), thanks to the lightweight world model and the guided search enabled by the action prior.

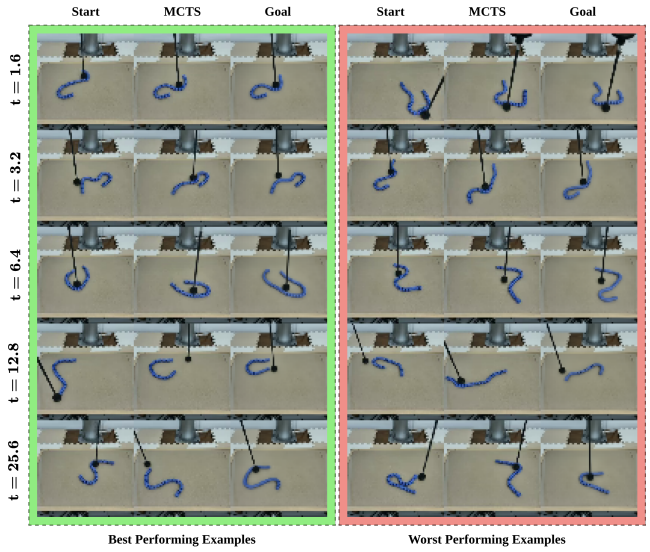

Fig. 8. The world model can predict complex deformations of the chain when pushed by the robot.

The MPC uses an unmasked DINOv2 distance to track plan states—sufficient given the small expected gap between observation and target during execution. Each control step optimizes two spline knot points over a 0.8 s horizon with K=16 samples and N_{MPC}=10 iterations, then executes a 0.4 s action chunk (Alg. 2). Fig. 5 shows real-robot tracking for three T-tool initializations.

V. Conclusion and Future Works

In this paper, we propose a framework that, instead of relying on a policy to memorize dataset actions as in behavior cloning, generates robot motion through planning and control in learned visual world models. Our robotic experiments demonstrate that visual world models, trained exclusively on a few hours of goal-independent play data, enable long-horizon forward simulation and serve as the foundation for MCTS and MPC planners to synthesize new trajectories. The

framework presents a promising direction for transferring knowledge across diverse tasks and supporting lifelong interaction experience. In future work, we aim to incorporate foundation joint action and world models to enable hierarchical planning and zero-shot task execution in novel settings. We also note that reward design remains an important practical consideration: while the masked DINOv2-based reward provides a general proof of concept for goal-image-conditioned tasks, its effectiveness depends on the quality of the visual embedding and the masking strategy. Recent work on general-purpose robotic reward learning [26] presents a promising direction toward more scalable and task-agnostic reward specification, and integrating such models within our planning framework is a natural avenue for future investigation.

REFERENCES

[1] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[3] H. Zhu, T. Zhao, X. Ni, J. Wang, K. Fang, L. Righetti, and T. Pang, "Should we learn contact-rich manipulation policies from sampling-based planners?" *IEEE Robotics and Automation Letters*, 2025.

[4] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," *Advances in neural information processing systems*, vol. 35, pp. 26 565–26 577, 2022.

[5] E. Alonso, A. Jelley, V. Micheli, A. Kanervisto, A. J. Storkey, T. Pearce, and F. Fleuret, "Diffusion for world modeling: Visual details matter in atari," *Advances in Neural Information Processing Systems*, vol. 37, pp. 58 757–58 791, 2024.

[6] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse control tasks through world models," *Nature*, pp. 1–7, 2025.

[7] N. Hansen, H. Su, and X. Wang, "Td-mpc2: Scalable, robust world models for continuous control," 2024.

[8] N. A. Hansen, H. Su, and X. Wang, "Temporal difference learning for model predictive control," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 8387–8406. [Online]. Available: https://proceedings.mlr.press/v162/hansen22a.html

[9] I. Georgiev, V. Giridhar, N. Hansen, and A. Garg, "Pwm: Policy learning with large world models," *arXiv preprint arXiv:2407.02466*, 2024.

[10] N. Agarwal, A. Ali, M. Bala, Y. Balaji, E. Barker, T. Cai, P. Chattopadhyay, Y. Chen, Y. Cui, Y. Ding *et al.*, "Cosmos world foundation model platform for physical ai," *arXiv preprint arXiv:2501.03575*, 2025.

[11] M. Assran, A. Bardes, D. Fan, Q. Garrido, R. Howes, M. Muckley, A. Rizvi, C. Roberts, K. Sinha, A. Zholus *et al.*, "V-jepa 2: Self-supervised video models enable understanding, prediction and planning," *arXiv preprint arXiv:2506.09985*, 2025.

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

[13] C. Neary, O. G. Younis, A. Kuramshin, O. Aslan, and G. Berseth, "Improving pre-trained vision-language-action policies with model-based search," 2025. [Online]. Available: https://arxiv.org/abs/2508.12211

[14] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[15] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel, "Daydreamer: World models for physical robot learning," *Conference on Robot Learning*, 2022.

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[17] H. Zheng, J. Fu, Y. Zeng, J. Luo, and Z.-J. Zha, "Learning semantic-aware normalization for generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 853–21 864, 2020.

[18] J. Bruce, M. D. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps *et al.*, "Genie: Generative interactive environments," in *Forty-first International Conference on Machine Learning*, 2024.

[19] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[20] D. Yang, D. Tjia, J. Berg, D. Damen, P. Agrawal, and A. Gupta, "Rank2reward: Learning shaped reward functions from passive video," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 2806–2813.

[21] A. Jordana, J. Zhang, J. Amigo, and L. Righetti, "An introduction to zero-order optimization techniques for robotics," *arXiv preprint arXiv:2506.22087*, 2025.

[22] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," *arXiv preprint arXiv:2212.00541*, 2022.

[23] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, S. Palma, P. Kooijmans, M. Aractingi, M. Shukor, D. Aubakirova, M. Russi, F. Capuano, C. Pascal, J. Choghari, J. Moss, and T. Wolf, "Lerobot: State-of-the-art machine learning for real-world robotics in pytorch," https://github.com/huggingface/lerobot, 2024.

[24] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[25] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[26] A. Liang, Y. Korkmaz, J. Zhang, M. Hwang, A. Anwar, S. Kaushik, A. Shah, A. S. Huang, L. Zettlemoyer, D. Fox *et al.*, "Robometer: Scaling general-purpose robotic reward models via trajectory comparisons," *arXiv preprint arXiv:2603.02115*, 2026.