FLEXIBLE LOCOMOTION LEARNING WITH DIFFUSION MODEL PREDICTIVE CONTROL

Runhan Huang[‡], **Haldun Balim**[‡], **Heng Yang**[‡], **Yilun Du**[‡†]
Flexible-Diffusion-MPC.github.io/ [*]

ABSTRACT

Legged locomotion demands controllers that are both robust and adaptable, while remaining compatible with task and safety considerations. However, model-free reinforcement learning (RL) methods often yield a *fixed* policy that can be difficult to adapt to new behaviors at test time. In contrast, Model Predictive Control (MPC) provides a natural approach to flexible behavior synthesis by incorporating different objectives and constraints directly into its optimization process. However, classical MPC relies on accurate dynamics models, which are often difficult to obtain in complex environments and typically require simplifying assumptions. We present **Diffusion-MPC**, which leverages a learned generative diffusion model as an approximate dynamics prior for planning, enabling flexible test-time adaptation through reward and constraint based optimization. Diffusion-MPC jointly predicts future states and actions; at each reverse step, we incorporate reward planning and impose constraint projection, yielding trajectories that satisfy task objectives while remaining within physical limits. To obtain a planning model that adapts beyond imitation pretraining, we introduce an interactive training algorithm for diffusion based planner: we execute our reward-and-constraint planner in environment, then filter and reweight the collected trajectories by their realized returns before updating the denoiser. Our design enables strong test-time adaptability, allowing the planner to adjust to new reward specifications without retraining. We validate Diffusion-MPC on real world, demonstrating strong locomotion and flexible adaptation.

1 INTRODUCTION

Legged locomotion remains a significant challenge in robotics, as controllers must guarantee stability under complex contact-rich dynamics (Kumar et al., 2021; Lee et al., 2020) while accommodating evolving task requirements. A generalized controller demands the synthesis of diverse behaviors—ranging from posture adjustment (Margolis & Agrawal, 2023; Ai et al., 2025) and balancing (Zhang et al., 2025) to energy-efficient walking (Nai et al., 2025). This notion of flexible behavior synthesis (Janner et al., 2022) is central to enabling robots to operate autonomously in unpredictable environments. Rather than being limited to a single, pre-defined policy, a capable locomotion controller should be able to integrate new constraints, and adapt to novel objectives on the fly (Janner et al., 2022). Such flexibility is especially critical at test time, where robots may face new terrains, altered physical limits, or modified task goals, which can not be anticipated during training or even design phase. Meeting this demand requires not only robustness to disturbances but also rapid test-time adaptation—the ability to reshape trajectories in real time.

Building toward this vision, the field has converged on two main solutions to locomotion control. Model predictive control (Rawlings et al., 2017) offers a natural mechanism for adaptation by explicitly encoding objectives and constraints into an optimization-based control framework. However, MPC depends on highly accurate dynamics models, incurs significant computational cost in contact-rich settings, and in practice often requires extensive simplifications and hand-crafted design choices (Bledt et al., 2018; Neunert et al., 2018). By contrast, model-free reinforcement learning (Schulman et al., 2017; Haarnoja et al., 2018) has achieved impressive results for behaviors (Kumar et al., 2021;

[*‡]Kempner Institute, Harvard University, MA, USA
[††] Corresponding at: ydu@seas.harvard.edu

Zhuang et al., 2023; Margolis & Agrawal, 2023), producing robust policies that can transfer from simulation to hardware; yet these methods typically learn a fixed policy network, making it difficult to flexibly synthesize multiple behaviors at test time and thus most successful in relatively static tasks. Together, these limitations highlight the gap between RL's efficiency and MPC's flexibility, motivating approaches that combine their respective strengths.

We address this gap by introducing **Diffusion-MPC**, which leverages generative diffusion models (Ho et al., 2020; Janner et al., 2022; Song & Ermon, 2019; Song et al., 2020) as expressive learned priors, enabling them to function as planners that implicitly capture system dynamics. Rather than directly fitting an action only policy, our diffusion model learns to jointly represent state transitions and action proposals from large, heterogeneous datasets. This learned generative prior then plays the role of the planner in an MPC framework: during each planning cycle, trajectories are sampled from the diffusion model and optimized with reward terms and constraints, effectively performing model-based planning without reliance on hand-crafted dynamics. In this view, diffusion models are not just conditional generators, but expressive approximators of environment dynamics that make tractable, flexible MPC possible. Reward-based planning updates steer generated trajectories toward task objectives, while feasibility is maintained through constraint projection. Candidate ranking is then applied to further refine the selected plan. Together, these mechanisms provide adaptability while avoiding the need for simplified model designs required by MPC and the rigidity of fixed RL policies. In addition, the framework naturally supports skill compositionality: multiple reward and constraint terms can be combined at test time to synthesize behaviors without retraining.

Directly training a diffusion planner on demonstrations, however, often fails to meet deployment requirements. In practice, datasets not only may lack coverage of critical objectives such as energy efficiency or smoothness, but may also suffer from limited quality or inconsistencies, since there is rarely access to a true expert capable of providing optimal demonstrations (Cao & Sadigh, 2021). Furthermore, demonstrations are typically collected in simulation under simplified physics, whereas deployment occurs in the real world where conditions such as terrain inclination, contact friction, and unmodeled disturbances differ substantially (Kumar et al., 2021). As a result, purely imitative rollouts may drift off-distribution during real (Huang et al., 2024). To address this, we introduce an interactive online training procedure for diffusion based planners: the diffusion planner is rolled out interactively to collect trajectories, and the model is updated using trajectories filtered and reweighted by realized returns, in the spirit of reward-weighted regression (Peters & Schaal, 2007; Kang et al., 2023; Peng et al., 2019; Ren et al., 2024). This adaptation both enhances robustness and improves alignment with task objectives, while preserving the compositional interface for flexible test-time control.

In summary, to tackle the challenge of achieving flexible and adaptable locomotion, we propose a novel framework that makes the following key contributions:

- **Diffusion-MPC Formulation.** We reinterpret diffusion models as generative priors for model predictive control via reward-based updates, constraint projection, and candidate ranking.
- **Interactive Training.** We propose a reward-weighted denoising adaptation procedure that finetunes the diffusion planner during interaction with the environment, enhancing locomotion capability.
- **Real-World Deployment.** We design practical techniques for real-time diffusion planning, including asynchronous execution and early-step caching, and demonstrate zero-shot transfer on a Unitree Go2 quadruped.

2 RELATED WORKS

2.1 LEARNING-BASED LOCOMOTION

Two main paradigms dominate locomotion control: model predictive control and reinforcement learning. MPC approaches for legged locomotion (Bledt et al., 2018; Neunert et al., 2018) optimize trajectories with explicit costs and constraints, enabling agile and versatile maneuvers. However, these methods typically rely on simplified dynamics models, which limits accuracy in contact-rich locomotion and makes them computationally demanding, often necessitating substantial model-

ing simplifications. By contrast, model-free RL (Schulman et al., 2017; Haarnoja et al., 2018) has achieved impressive results through massively parallel simulation training (Makoviychuk et al., 2021; Gu et al., 2023; Tao et al., 2024). The learned policies was successfully deployed in different scenarios, including traversing complex terrains (Kumar et al., 2021; Wu et al., 2023; Zhu et al., 2025a; Cheng et al., 2024b; Ren et al., 2025), achieving extreme motions (He et al., 2024; Zhang et al., 2025; Huang et al., 2025a; Zhuang et al., 2023; Luo et al., 2024a; Cheng et al., 2024a; Hoeller et al., 2024), and interactively navigate in the real world (Zhang et al., 2024; Zhu et al., 2025c;b; Uppal et al., 2024). Yet policies learned via model-free RL are tightly coupled to their training reward functions, confining them to predefined behaviors and hindering adaptation to novel test-time objectives. Hence the generalizability and adaptability of RL is not desirable for flexible behavior synthesis. Works such as (Margolis & Agrawal, 2023) further shows that reinforcement learning can endow a single policy with a family of behaviors and interpolate between them. However, the set of behaviors must be predefined through task-specific reward design and integrate into network input during training, limiting flexibility when new objectives arise at deployment. Building on the powerful expressiveness of generative models (Ho et al., 2020; Song et al., 2020), recent approaches (Chi et al., 2023; Janner et al., 2022) have explored their use as planners that capture system dynamics implicitly. In this view, diffusion models provide a learned generative prior over trajectories, which can then be steered by reward terms and constraints at test time to enable flexible adaptation.

2.2 Diffusion for Control

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Bansal et al., 2023; Black et al., 2023; Peebles & Xie, 2023) have recently been explored as generative frameworks for decision making (Chi et al., 2023; Janner et al., 2022; Ajay et al., 2022; Qi et al., 2025; Römer et al., 2024; Chen et al., 2024; Luo et al., 2025; Karunratanakul et al., 2023; Huang et al., 2025b; Luo et al., 2024b). Recently, there have been approaches to leverage the expressiveness of diffusion in locomotion control. Huang et al. (2024) use diffusion policy as action generator to learn from large corpus of demonstrations. O'Mahoney et al. (2025) adopt a classifier free guidance with return on the diffusion policy to control the robot toward different behavior, but still have to predefine one-hot skill vectors as network input and incorporate pre-defined return in training time, which limits the adaptability and flexibility of behavior. Zhou et al. (2025) and Huang et al. (2025b) has shown that diffusion models can serve as generative priors for planning, but the framework focused on eithor relatively simple planning or was only evaluated in simulation. Compared to the action-only diffusion motion generation (Huang et al., 2024; O'Mahoney et al., 2025), we incorporate implicit dynamics modeling through modeling state-action distribution and conduct planning during deployment, enabling flexible behavior at test-time.

3 Problem Setup and Background

We consider a discrete-time dynamical system $s_{t+1} = f(s_t, a_t)$, where $s_t \in \mathbb{R}^{n_s}$ denotes the state and $a_t \in \mathbb{R}^{n_a}$ the control input at time t. The objective is to maximize a cumulative reward $r(s_t, a_t)$ while ensuring that the trajectories satisfy constraints $(s_t, a_t) \in \mathcal{C}$.

Since optimizing over long horizons is typically computationally prohibitive, MPC (Rawlings et al., 2017) instead considers a finite and tractable horizon of length H. At each time step, MPC plans over this horizon by solving

$$\max_{a_{t:t+H-1|t}} \quad \sum_{i=0}^{H-1} r(s_{t+i|t}, a_{t+i|t})$$

subject to the dynamics constraint $s_{t+i+1|t} = f(s_{t+i|t}, a_{t+i|t})$ and the constraint $(s_{t+i|t}, a_{t+i|t}) \in \mathcal{C}$ for $i = 0, \ldots, H-1$, starting from the current state s_t. After solving this optimization, only the first action $a_{t|t}$ is applied, and the problem is re-solved at the next step in a receding-horizon fashion.

While classical MPC frameworks rely on solving an optimization problem at each step, a recently popular alternative is to cast planning as a conditional generation problem (Janner et al., 2022; Ajay et al., 2022). Concretely, we consider that we have access to a trajectory dataset $\mathcal{D} = \{\xi_i\}_{i=1}^{N}, \quad \xi_i = \{(s_t^i, a_t^i)\}_{t=0}^{T}$, collected from diverse policies that are not necessarily optimal nor guaranteed to satisfy the constraints in $\mathcal{C}$. From this dataset, we learn a generative model

that samples state–action sequences of horizon H:

$$\tau = \begin{bmatrix} s_t & s_{t+1} & \cdots & s_{t+H} \\ a_t & a_{t+1} & \cdots & a_{t+H} \end{bmatrix}, \quad \tau \in \mathbb{R}^{(n_s+n_a)\times H},$$

aimed at maximizing rewards while respecting constraints. Unlike optimization-based MPC, this sampling-based approach learns a generative prior over trajectories, allowing rewards and constraints to be incorporated directly during sampling.

Trajectory diffusion. Diffusion models (Sohl-Dickstein et al., 2015) provide a flexible generative framework by gradually perturbing data with noise and learning a reverse denoising process to recover structured samples. Beyond image and signal domains, they have also been successfully applied to trajectory modeling and control (Janner et al., 2022; Ajay et al., 2022). Motivated by these successes, we employ diffusion models to learn a generative prior over trajectories. Specifically, we consider a data-generating process:

$$q\Big(\tau^{(k)} \mid \tau^{(k-1)}\Big) = \mathcal{N}\Big(\sqrt{\alpha_k}\,\tau^{(k-1)},\ (1-\alpha_k)\,\mathbf{I}\Big),$$

and a learned reverse process:

$$p_\theta\Big(\tau^{(k-1)} \mid \tau^{(k)}\Big) = \mathcal{N}\Big(\mu_\theta(\tau^{(k)}, k),\ \Sigma_k\Big),$$

with a pre-defined variance schedule $\{\alpha\}_{k=0}^{K} \in (0, 1]$ and a neural network μ_θ. To learn the parameters θ, we use the denoising objective (Ho et al., 2020):

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau,k,\epsilon} \left\|\tau - \tau_\theta(\tau^{(k)}, k)\right\|^2,$$

where $\tau^{(k)}$ is obtained from the forward noising process and $\epsilon \sim \mathcal{N}(0, I)$.

4 METHOD

4.1 Flexible Behavior Synthesis Through Sampling.

In this section, we present our methodology for synthesizing reward-maximizing, constraint-satisfying trajectories using our learned generative prior. Given a differentiable trajectory-level reward function $R(\tau)$, we define the distribution of desired trajectories as

$$\pi(\tau) \propto p_\theta(\tau)\, \exp\big(\lambda R(\tau)\big)\, \mathbf{1}\{\tau \in \mathcal{C}\},$$

where $p_\theta(\tau)$ is the learned generative prior by our diffusion model, $\lambda \geq 0$ controls the strength of planning, and $\mathbf{1}\{\tau \in \mathcal{C}\}$ enforces constraints.

To sample from π, we tilt the diffusion prior $p_\theta(\tau)$ using reward-based planning, following the reward planning approach of (Janner et al., 2022). After each reverse step, we apply a reward planning update:

$$\tau^{(k-1)} := \tau^{(k-1)} + \eta_k\, \Sigma_k\, \nabla_\tau R\big(\tau^{(k-1)}\big),$$

where Σ_k is the covariance of the reverse transition and η_k controls the strength of planning. For the trajectory-level reward $R(\tau)$, we combine two complementary signals: a learned neural component R_{nn} that captures long-horizon semantic behaviors difficult to formalize analytically, and an analytic component R_{an} that encodes hand-specified objectives.

$$R(\tau) = \alpha_{\mathrm{nn}} R_{\mathrm{nn}}(\tau) + \alpha_{\mathrm{an}} R_{\mathrm{an}}(\tau).$$

This combination offers flexibility: neural rewards provide expressive, data-driven semantics, while analytic rewards enable direct incorporation of task-specific structure. Together, they allow planning to balance learned behavior with explicitly defined objectives. To ensure constraint, we project the denoised sample onto the feasible set after each step in reverse chain:

$$\tau^{(k-1)} := \Pi_{\mathcal{C}}\Big(\tau^{(k-1)}\Big),$$

where $\Pi_{\mathcal{C}}$ denotes the projection operator onto $\mathcal{C}$. This mechanism provides a flexible way to enforce constraint, ensuring that sampled trajectories remain valid while allowing diverse behaviors to emerge.

The overall sampling procedure is shown in Algorithm 1. Starting from Gaussian noise, the reverse process iteratively refines a candidate trajectory. At each step, we incorporate reward functions to plan the trajectory distribution toward high-reward behaviors and enforce constraint through projection. The initial state is fixed throughout the process to ensure consistency with the current observation. This combination of denoising and planning yields state–action trajectories that both adhere to constraints and flexibly adapt to task objectives.

Algorithm 1 Diffusion-MPC Sampling

Require: diffusion model μ_θ, reward planning scale λ, reward function $R(\tau)$, constraint set $\mathcal{C}$, initial state s_0, number of candidates N

All updates below are applied in parallel to the N candidates.
initialize $\tau^K \sim \mathcal{N}(0, I)$ with batch size N; set first state $\tau^K_{s_0} \leftarrow s_0$
for $k = K, \ldots, 1$ **do**
  $\tau^{k-1} \leftarrow \mu_\theta(\tau^k, k) + \sigma_k \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(0, I)$ ▷ reverse step
  $\tau^{k-1} \leftarrow \tau^{k-1} + \lambda \, \Sigma_k \, \nabla_\tau R(\tau^{k-1})$ ▷ reward planning
  $\tau^{k-1} \leftarrow \Pi_{\mathcal{C}}\big(\tau^{k-1}\big)$ ▷ constraint projection
  $\tau^{k-1}_{s_0} \leftarrow s_0$ ▷ enforce initial state
end for
$\tau^\star = \arg\max_{\tau \in \{\tau_i^0\}} R(\tau)$ ▷ ranking across N candidates

In addition to reward-based planning and constraint injection, we employ a candidate ranking strategy: multiple trajectories are sampled from the diffusion planning process, scored by the reward, and the best one selected. This complements local gradient updates by improving robustness through exploration of diverse rollouts. This global selection complements local gradient planning: while gradient steps provide fine-grained planning, candidate ranking improves robustness by exploring diverse rollouts and mitigating noisy gradients planning.

Compared with action-only diffusion approaches (Chi et al., 2023) that rely on separately conditioned dynamics models (Zhou et al., 2025), our joint state–action formulation provides a unified representation of trajectories and controls. By generating states and actions together, the diffusion prior naturally respects system dynamics, eliminating the need for an explicit dynamics model. Moreover, this formulation enables reward shaping and constraint injection to act directly on the evolving state sequence at every denoising step, rather than only through actions. This joint perspective increases flexibility in incorporating task objectives and feasibility constraints.

4.2 Compositional Behavior Synthesis

Beyond planning with a single reward function, our framework also supports flexible skill composition through reward combination. Let $\{R_i(\tau)\}_{i=1}^K$ denote a set of scalar task rewards, which may be either neural or analytic. At deployment, the user specifies weights $\alpha \in \mathbb{R}^K$ to form a composite objective $R_\alpha(\tau) = \sum_{i=1}^K \alpha_i \, R_i(\tau)$. By varying the weights α_i, Diffusion MPC can seamlessly trade off between different objectives, synthesizing a diverse range of behaviors. This includes not only behaviors represented in the dataset but also novel behaviors arising from new combinations of reward signals.

4.3 Planner Learning with Environment Interaction

We propose a strategy to collect data and finetune the diffusion prior using trajectories generated by our model in an online interactive way. Let $(\tau^{(k)}, k, \epsilon)$ denote a standard denoising tuple constructed from a clean trajectory τ with forward noise $\epsilon \sim \mathcal{N}(0, I)$. Per-trajectory weights are defined from realized returns as

$$w(R_r(\tau)) = \exp\left(\frac{R_r(\tau)}{T}\right),$$

with $T > 0$ as a temperature parameter. Here R_r denotes the ground-truth return from the environment, analogous to the RL setting, rather than the reward model used during planning. To filter out low-return rollouts, we retain only the top-K weights and set the rest to zero:

$$w'(\tau) = \begin{cases} w(R_r(\tau)), & \tau \in \text{Top-}K, \\ 0, & \text{otherwise.} \end{cases}$$

$$\bar{w}(\tau) = \frac{w'(\tau)}{\mathbb{E}[w'(\tau)]}.$$

The resulting objective is

$$\mathcal{L}_{\text{RWD}}(\theta) = \mathbb{E}\Big[\ \bar{w}(R_r(\tau))\ \Big\|\ \tau - \tau_\theta(\tau^{(k)}, k)\Big\|_2^2\Big].$$

which performs exponentially tilted regression, biasing updates toward higher-return trajectories. A replay buffer is maintained that interleaves on-policy planner rollouts with previously collected trajectories, preserving the coverage of prior experiences while nudging the model toward reward-favored regions of the trajectory space.

Compared with standard policy-gradient methods, reward-weighted denoising has several advantages for diffusion planners: (i) it remains in the native denoising parameterization, avoiding high-variance policy gradients and critic bootstrapping; (ii) it is naturally off-policy and sample-efficient—filter and reweight can exploit stored trajectories without importance-correction instabilities; (iii) it is flexible to explicitly plan into several behavior in the sampling phase, giving strong guidance during the rollout. In practice, this yields a stable online procedure that trains a planner flexibly while preserving the benefits of the learned generative prior.

4.4 Real-time Planning

Asynchronous planning for real-time control. To meet high-rate locomotion requirements, we employ an asynchronous pipeline with planning horizon H and replan margin D. At each timestep t, the controller executes the next action from the current H-step plan $a_{t:t+H-1}$. When the execution index reaches $H-D$, we trigger replanning from the latest observation to synthesize a fresh H-step plan while continuing to execute the remaining D buffered actions from the old plan. Once these D actions have been applied, we time-align the new plan by skipping its first D actions and begin execution at offset D. Equivalently, each action is computed D control cycles before it is applied (a D-step action buffer), which maintains real-time operation while preserving closed-loop feedback with period $H-D$ steps. In our experiments, we set H=11 and D=3.

Caching early denoising. Successive plans generated by our model often produce nearly identical trajectories in early diffusion steps, as these steps primarily denoise without incorporating task-specific structure. To avoid redundant computation, we shift the existing plan across time steps and reuse it as the initialization for the next window, up to m steps. This warm-start strategy preserves solution quality while substantially reducing inference cost.

Sampler choice and step budget. DDIM offers faster, deterministic sampling at some cost in fidelity, while DDPM is slower but higher quality. The number of denoising steps controls the compute–quality trade-off. We use 10 DDPM steps at inference and ablate both the training horizon and test-time step count.

5 Experiments

5.1 Experiment Setup

Adaptation Tasks. For adaptation tasks, we consider locomotion tasks with objectives: base height variation, joint limit restriction, energy saving, joint acceleration/velocity regularization, and balancing. Detailed task definition and reward formulations can be found in AppendixA. We use Isaacgym (Makoviychuk et al., 2021) as our simulator for pretrain dataset collection and interactive training. The planners are initialized with dataset collected with domain randomization. Our interactive diffusion planner training in 5.3 is conducted on a single NVIDIA 3090 GPU, with 4096 environments in simulation. The control frequency in both the simulation environment and the real world is 50Hz. Our policy is deployed on a Unitree Go2 quadrupedal robot, with an Intel NUC 12 PRO as onboard computing device. We use PD control for low-level joint torques ($K_p = 40.0, K_d = 1.0$). The robot observation at time t is represented as

$$o_t = \{\ v_t^{\text{yaw}},\ g_t,\ v_t^{\text{cmd}},\ q_t,\ \dot{q}_t,\ a_{t-1}\ \},$$

where v_t^{yaw} is the base angular velocity, g_t is the projected gravity, v_t^{cmd} is the command vector, q_t and $\dot{q}_t$ are the joint position and velocity, and a_{t-1} is the previous action.

Interactive Learning Setup. Our training follows a two-stage procedure. In the first stage, we pretrain the planner on $4{,}000$ trajectories of length $1{,}000$, collected from a demonstrator policy trained with PPO. The demonstration data is collected in a static environment which is not sufficiently representative for robust deployment. To address this, we finetune the planner as described in Sec. 4.3, using 1500 additional environment interactions for each environments. During this stage, we apply domain randomization with the parameters detailed in Tab. 3.

5.2 Simulation Experiments

Objectives	Ref. Diff.	Baseline (No Reward/Constraint)			Reward Planning (Reward Only)			Constr. (Only)
	Policy	C1	C10	C100	C1	C10	C100	C1
Joint Vel	0.969	0.973	0.752	0.743	0.762	0.697	0.684	**0.553**
Joint Acc	0.350	0.349	0.263	0.254	0.267	0.259	**0.249**	0.262
Joint Pos (N)	0.051	0.484	0.348	0.343	0.296	0.317	0.307	**0.183**
Joint Pos (P)	0.243	0.240	0.238	0.237	0.209	0.208	0.204	**0.168**
Balancing	0.807	0.768	0.663	0.650	0.562	0.564	**0.551**	0.662
Energy	0.774	0.825	0.664	0.626	0.519	0.498	**0.482**	0.700

Table 1: **Adaptation Performance.** Normalized penalties (lower is better). We compare the Diffusion Policy against Sampling methods with varying candidate counts (C1/10/100) and planning components (Reward/Constraint).

Adaptation Tasks. The adaptation capability of Diffusion-MPC is assessed across a range of objectives using 1,000 environments over 3,000 simulation steps, with the average penalty reported as the evaluation metric. The study systematically varies the number of candidate trajectories together with the inclusion of reward-based planning and constraint projection to examine their individual and combined contributions. Joint Pos (N) refers to tasks with negative rear-leg joint position targets, while Joint Pos (P) corresponds to positive targets. To ensure fair assessment, both linear and angular velocity tracking are maintained within 97% of the baseline policy without planning, preventing excessive trade-off between task adaptation and nominal locomotion quality. Tracking performance is quantified using an exponential error metric, where the squared difference between commanded and realized velocities is penalized and mapped through $\exp(-\|e\|^2/\sigma)$, yielding values close to one for accurate tracking and approaching zero otherwise.

Table 1 demonstrates the adaptation ability of Diffusion-MPC across variations in candidate number, reward planning, and constraint planning. All values are normalized penalties, with less penalty denoting better alignment to desired behaviors. Increasing the candidate number consistently improves performance by enabling more diverse trajectory proposals. Reward planning further reduces penalties across all task categories, demonstrating that reward planning provides broadly beneficial trajectory shaping beyond candidate diversity. Constraint projection is most critical for feasibility-dominated tasks such as joint position adaptation, directly enforcing safe joint configurations. The penalty remains nonzero not due to failure, but because constraints intentionally extend beyond the demonstration distribution; in these regions, our planner successfully generalizes to discover relatively feasible actions consistent with the planning objective, though the resulting penalty is not exactly zero.

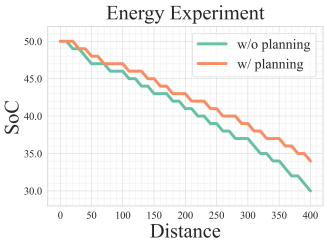
Interactive Learning Experiments. We compare a planner initialized on offline data and subsequently finetuned via interactive training against a baseline initialized on the same offline data without finetuning. Each configuration is evaluated in 1000 parallel environments for 1000 steps, and we report success rate and mean velocity deviation. As shown in Table 1a, finetuning improves stability and velocity-tracking accuracy, with the largest gains at high commanded speeds.

5.3 Real-world Experiments

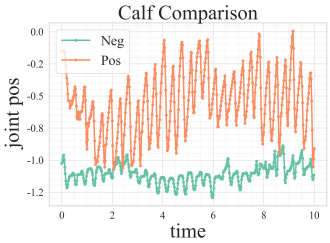
Adaptation Tasks. Adaptation capability is evaluated across four representative tasks: energy saving, joint position regulation, height variation, and dynamic balancing.

Speed	Metric	Ours	Baseline
0.5 m/s	Stability (%)	**99.7**	46.6
	E_v (%)	**29.6**	51.4
0.7 m/s	Stability (%)	**90.5**	23.1
	E_v (%)	**28.5**	77.0
1.0 m/s	Stability (%)	**98.8**	18.4
	E_v (%)	**29.1**	74.8

(a) Simulation Finetuning



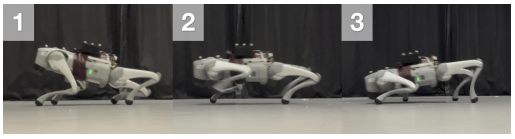
(b) State of Charge



(c) Left Rear Calf Position

Figure 1: **Experimental Results.** (a) Simulation finetuning results (stability rate and velocity error E_v at different speeds); (b) SoC during the long-distance real-world evaluation; (c) Left rear calf position under different planning settings.

(a) **Height transition** experiment.



(b) **Different joint position targets** experiment.

Figure 2: Height variation and joint position adaptation experiments.

For energy saving task, we record the state of charge (SoC) of the robot every 10 meters. We initiate the robot to 50% SoC, and command the robot to walk with 0.6 m/s. As shown in Fig.1, the overall energy saving is 20% with energy saving reward compared to no planning. The energy saving oriented agent demonstrate a smoother action pattern and decrease the foot lifting height to avoid unnecessary energy consumption that doesn't contribute to the actual movement.

For joint position adaptation, two distinct reference joint poses are provided for the robot to track, which differ from the patterns of the source RL policy. The *negative* setting enforces a relatively small calf angle (close to extension), while the *positive* setting requires a larger angle (more flexed). Figure 1 shows the resulting rear-left calf trajectories. Under the positive target, the calf maintains a higher average position with large oscillations, indicating active participation in locomotion. In contrast, under the negative target the calf remains closer to extension with small oscillations, contributing less to propulsion. Figure 2b further illustrates the resulting strategies: the negative target induces an upward tilt with propulsion dominated by the front legs, while the positive target shifts the robot forward, with larger calf–thigh angles and the front calves primarily used for balance.

For height variation, a neural-network-based height reward model directs the robot from a relatively elevated base position to a lower one. As shown in Fig. 2a, the robot transitions from an initial height of 25 cm to a reduced height of 18 cm under reward planning. It is important to note that while the pretraining dataset includes demonstrations at both heights, it does not contain any demonstrations of transitions between them; the observed adaptation is therefore entirely achieved through test-time planning.

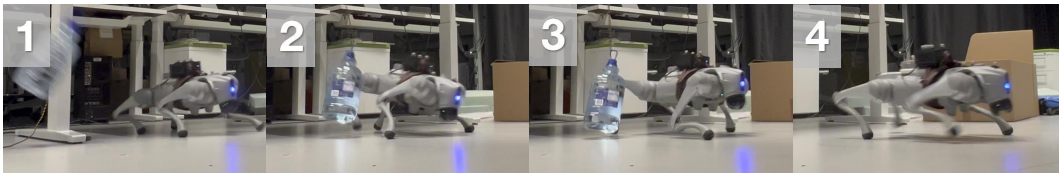


Figure 3: **Balancing under external disturbance.** The robot is subjected to a lateral push at the trunk and subsequently recovers from the perturbed posture, reestablishing balance.

For balancing, a pendulum apparatus is employed to generate controlled lateral impacts, similar to (Xiao et al., 2024). As shown in Fig. 3, a 3.8 kg weight suspended from a 1.2 m pivot swings to strike the robot's trunk at its lowest point, producing impact velocities of 4.54 m/s (90° release) and 3.21 m/s (60° release), corresponding to two difficulty levels. A trial is considered a failure if the robot falls or remains stuck below 0.2 m/s for more than 2 s. As summarized in Table 2a, incorporating balancing rewards and constraint planning markedly improves the ability to withstand external disturbances and aids recovery.

	Easy	Medium	Hard
No Plan	0.9	0.8	0.6
Plan	1.0	1.0	0.9

(a) Balancing success rates

Task (Speed)	Finetuned	Baseline
0.5 m/s	**1.0**	0.2
0.7 m/s	**1.0**	0.1
1.0 m/s	**1.0**	0.0

(b) Real-world finetuning rates

Table 2: **Experimental success rate statistics.** (a) Balancing success rate under different pendulum angles (Easy/Medium/Hard); (b) Real-world finetuning success rates at different speeds, where the finetuned planner outperforms the baseline significantly.

Interactive Learning. As shown in Fig. 4, both the planner trained on datasets without domain randomization and its finetuned counterpart are evaluated in real-world experiments. As shown in Table 2b, the robot is tested at different target speeds over a 10 m trajectory, and success rates are reported. The finetuned policy exhibits substantial gains over the untuned version. By contrast, the untuned policy performs even worse than in simulation, likely due to the replan margin during deployment, which reduces responsiveness. In addition, Fig. 5 presents real-world deployment of a planner trained entirely from scratch, demonstrating that diffusion-based planning can be learned effectively without reliance on demonstration data.

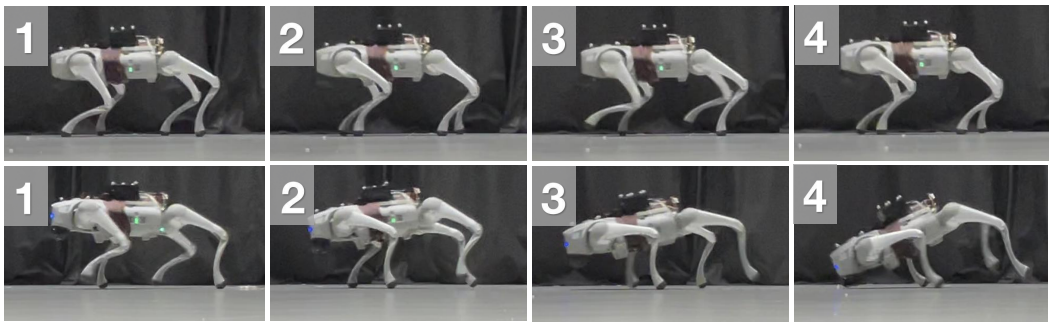


Figure 4: **Comparison of finetuned and untuned diffusion planners in real-world deployment.** The finetuned planner (top) achieves stable locomotion, while the untuned baseline (bottom) fails to deploy.

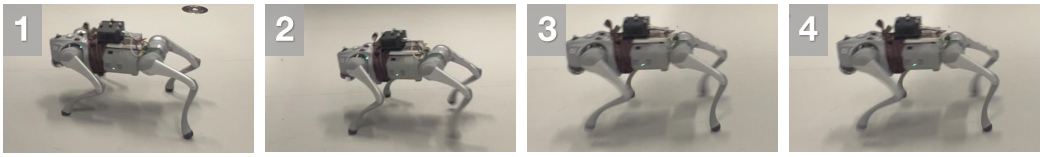


Figure 5: **Interactive diffusion planner learned from scratch** deployed into real world

6 CONCLUSION

We present Diffusion-MPC, a planning framework that leverages diffusion models as expressive generative priors over state–action trajectories, synthesizing generalized behaviors driven by a broad range of rewards and constraints. Diffusion-MPC supports novel reward functions and constraints at test time, allowing flexible adaptation to new tasks and environments. Our approach leverages the dataset to capture and compose existing behaviors into novel ones, and further employs an online finetuning mechanism that actively explores the task space to recover behaviors absent from demonstrations, enhancing overall performance. Diffusion-MPC runs in real time via asynchronous planning and early-step caching, and exhibits flexible behavior adaptation. Our work positions diffusion-based generative priors as a practical path to adaptable, general-purpose embodied control.

Future directions include extending the framework to integrate diverse inputs—such as LiDAR, visual perception, and natural language—advancing toward embodied agents capable of fully autonomous real-world interaction and planning. Our present study considers only simple constraint classes; extending the approach to richer, task-dependent (and potentially nonconvex or time-varying) constraints is a natural next step. Another key direction is to develop interactive finetuning methods that can train competitive planners without relying on offline data, thereby removing the need for expert demonstrations while preserving sample efficiency.

REFERENCES

Bo Ai, Liu Dai, Nico Bohlinger, Dichen Li, Tongzhou Mu, Zhanxin Wu, K Fay, Henrik I Christensen, Jan Peters, and Hao Su. Towards embodiment scaling laws in robot locomotion. *arXiv preprint arXiv:2505.05753*, 2025.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 843–852, 2023.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

Gerardo Bledt, Matthew J Powell, Benjamin Katz, Jared Di Carlo, Patrick M Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2245–2252. IEEE, 2018.

Zhangjie Cao and Dorsa Sadigh. Learning from imperfect demonstrations from agents with varying dynamics. *IEEE Robotics and Automation Letters*, 6(3):5231–5238, 2021.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11443–11450. IEEE, 2024a.

Yi Cheng, Hang Liu, Guoping Pan, Houde Liu, and Linqi Ye. Quadruped robot traversing 3d complex environments with limited perception. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9074–9081. IEEE, 2024b.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, pp. 02783649241273668, 2023.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Tairan He, Chong Zhang, Wenli Xiao, Guanqi He, Changliu Liu, and Guanya Shi. Agile but safe: Learning collision-free high-speed legged locomotion. *arXiv preprint arXiv:2401.17583*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.

Runhan Huang, Shaoting Zhu, Yilun Du, and Hang Zhao. Moe-loco: Mixture of experts for multitask locomotion. *arXiv preprint arXiv:2503.08564*, 2025a.

Xiaoyu Huang, Yufeng Chi, et al. Diffuseloco: Real-time legged locomotion control with diffusion from offline datasets. In *8th Annual Conference on Robot Learning*, 2024.

Xiaoyu Huang, Takara Truong, Yunbo Zhang, Fangzhou Yu, Jean Pierre Sleiman, Jessica Hodgins, Koushil Sreenath, and Farbod Farshidian. Diffuse-cloc: Guided diffusion for physics-based character look-ahead control. *ACM Transactions on Graphics (TOG)*, 44(4):1–12, 2025b.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 17–23 Jul 2022.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

Korrawe Karunratanakul, Konpat Preechakul, Supasorn Suwajanakorn, and Siyu Tang. Guided motion diffusion for controllable human motion synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2151–2162, 2023.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

Shixin Luo, Songbo Li, Ruiqi Yu, Zhicheng Wang, Jun Wu, and Qiuguo Zhu. Pie: Parkour with implicit-explicit learning framework for legged robots. *IEEE Robotics and Automation Letters*, 2024a.

Yunhao Luo, Chen Sun, Joshua B Tenenbaum, and Yilun Du. Potential based diffusion motion planning. *arXiv preprint arXiv:2407.06169*, 2024b.

Yunhao Luo, Utkarsh A Mishra, Yilun Du, and Danfei Xu. Generative trajectory stitching through diffusion composition. *arXiv preprint arXiv:2503.05153*, 2025.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Conference on Robot Learning*, pp. 22–31. PMLR, 2023.

Ruiqian Nai, Jiacheng You, Liu Cao, Hanchen Cui, Shiyuan Zhang, Huazhe Xu, and Yang Gao. Fine-tuning hard-to-simulate objectives for quadruped locomotion: A case study on total power saving. *arXiv preprint arXiv:2502.10956*, 2025.

Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, 2018.

Reece O'Mahoney, Alexander L Mitchell, Wanming Yu, Ingmar Posner, and Ioannis Havoutis. Offline adaptation of quadruped locomotion using diffusion models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9974–9980. IEEE, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Strengthening generative robot policies through predictive world modeling. *arXiv preprint arXiv:2502.00622*, 2025.

James B Rawlings, David Q Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

Junli Ren, Tao Huang, Huayi Wang, Zirui Wang, Qingwei Ben, Junfeng Long, Yanchao Yang, Jiangmiao Pang, and Ping Luo. Vb-com: Learning vision-blind composite humanoid locomotion against deficient perception. *arXiv preprint arXiv:2502.14814*, 2025.

Ralf Römer, Alexander von Rohr, and Angela P Schoellig. Diffusion predictive control with constraints. *arXiv preprint arXiv:2412.09342*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

Shagun Uppal, Ananye Agarwal, Haoyu Xiong, Kenneth Shaw, and Deepak Pathak. Spin: Simultaneous perception interaction and navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18133–18142, 2024.

Jinze Wu, Guiyang Xin, Chenkun Qi, and Yufei Xue. Learning robust and agile legged locomotion using adversarial motion priors. *IEEE Robotics and Automation Letters*, 2023.

Zhiyuan Xiao, Xinyu Zhang, Xiang Zhou, and Qingrui Zhang. Pa-loco: Learning perturbation-adaptive locomotion for quadruped robots. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9110–9115. IEEE, 2024.

Chong Zhang, Jin Jin, et al. Resilient legged local navigation: Learning to traverse with compromised perception end-to-end. In *ICRA*, pp. 34–41, 2024.

Tong Zhang, Boyuan Zheng, Ruiqian Nai, Yingdong Hu, Yen-Jen Wang, Geng Chen, Fanqi Lin, Jiongye Li, Chuye Hong, Koushil Sreenath, et al. Hub: Learning extreme humanoid balance. *arXiv preprint arXiv:2505.07294*, 2025.

Guangyao Zhou, Sivaramakrishnan Swaminathan, Rajkumar Vasudeva Raju, J Swaroop Guntupalli, Wolfgang Lehrach, Joseph Ortiz, Antoine Dedieu, Miguel Lazaro-Gredilla, and Kevin Patrick Murphy. Diffusion model predictive control. *Trans. on Mach. Learning Research*, 2025. ISSN 2835-8856.

Shaoting Zhu, Runhan Huang, Linzhan Mou, and Hang Zhao. Robust robot walker: Learning agile locomotion over tiny traps. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15987–15993. IEEE, 2025a.

Shaoting Zhu, Derun Li, Linzhan Mou, Yong Liu, Ningyi Xu, and Hang Zhao. Saro: Space-aware robot system for terrain crossing via vision-language model. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14820–14827. IEEE, 2025b.

Shaoting Zhu, Linzhan Mou, Derun Li, Baijun Ye, Runhan Huang, and Hang Zhao. Vr-robo: A real-to-sim-to-real framework for visual robot navigation and locomotion. *IEEE Robotics and Automation Letters*, 2025c.

Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher Atkeson, Soeren Schwertfeger, Chelsea Finn, and Hang Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023.

A Adaptation Task Definition

A.1 Base Height Variation

Base height is not included in the observation. We therefore use a trajectory-level reward model

$$R_{\text{height}}(\tau;\, h^\star) \;=\; f_\phi\big(\tau^{(k)},\, t\big)\,,$$

where a U-Net takes a corrupted joint state–action segment $\tau^{(k)}$ and diffusion step t and predicts the reward of the corresponding clean trajectory; at test time, $h^\star$ denotes the desired height profile implicit in the query. In our setting, we set $h^\star = 0.15$ m.

A.2 Joint Limit Restriction

We consider (i) attraction to a target posture q^{tar} via an analytic term

$$R_{\text{posture}}(\tau) \;=\; -\frac{1}{H}\sum_{t=0}^{H-1} \|q_t - q^{\text{tar}}\|_2^2,$$

and (ii) a tightened range $q^{\min} \le q_t \le q^{\max}$ enforced by reward

$$R_{\text{range}}(\tau) = -\frac{1}{H}\sum_{t=0}^{H-1}\Big(\big\|[\,q_t - q^{\max}\,]_+\big\|_2^2 + \big\|[\,q^{\min} - q_t\,]_+\big\|_2^2\Big)$$

and also projection

$$q_t \leftarrow \Pi_{[\,q^{\min},\, q^{\max}\,]}(q_t).$$

where $[u]_+ = \max(u, 0)$ and $\Pi_{[a,b]}(q) = \min(\max(q, a), b)$ acts elementwise.

A.3 Energy Saving

We train a reward model $R_{\text{energy}}(\tau) = f_\psi(\tau)$ to predict energy cost and plan accordingly. The reward is calculated by the time-integrated mechanical power

$$R_{\text{energy}}(\tau) \;=\; -\sum_{t=0}^{H-1}\sum_{j=1}^{d_u} \big|\tau_{j,t}\,\dot q_{j,t}\big|\,\Delta t,$$

with joint torque $\tau_{j,t}$ and velocity $\dot q_{j,t}$.

A.4 Joint Acceleration / Velocity Regularization

We penalize high rates with analytic terms

$$R_{\text{vel/acc}}(\tau) \;=\; -\lambda_v \sum_{t=0}^{H-1} \|\dot q_t\|_2^2 \;-\; \lambda_a \sum_{t=1}^{H-1} \left\| \frac{\dot q_t - \dot q_{t-1}}{\Delta t} \right\|_2^2,$$

and enforce rate limits by projection $\dot q_t \leftarrow \Pi_{[-\dot q^{\max},\, \dot q^{\max}]}(\dot q_t)$.

A.5 Balancing

We align the gravity direction measured in the body frame with a desired unit direction. Let $g_b(s_t)$ be the gravity vector expressed in the body frame (available from IMU/state), and define the unit vector $\hat g_t = g_b(s_t)/\|g_b(s_t)\|_2$. Let $\hat d_t$ be the desired unit gravity direction (for level posture, $\hat d_t = [0, 0, -1]^\top$). We use

$$R_{\text{align}}(\tau) = -\frac{1}{H}\sum_{t=0}^{H-1}\big(1 - \hat g_t^\top \hat d_t\big),$$

$$R_{\text{smooth}}(\tau) = -\lambda_{\text{tv}}\sum_{t=1}^{H-1}\|\hat g_t - \hat g_{t-1}\|_1,$$

$$R_{\text{balance}}(\tau) = R_{\text{align}}(\tau) + R_{\text{smooth}}(\tau).$$

B Interactive Learning Details

Our training follows a two-stage procedure. In the first stage, we pretrain the planner on $4{,}000$ trajectories of length $1{,}000$, collected from a demonstrator policy trained with PPO. The demonstration data is collected in a static environment which is not sufficiently representative for robust deployment. To address this, we finetune the planner as described in Sec. 4.3, using 1500 additional environment interactions for each environments. During this stage, we apply domain randomization with the parameters detailed in Tab. 3.

Parameters	**Range**	**Unit**
Base mass	$[1, 3]$	kg
Mass position (X)	$[-0.2, 0.2]$	m
Mass position (Y)	$[-0.1, 0.1]$	m
Mass position (Z)	$[-0.05, 0.05]$	m
Friction	$[0, 2]$	-
Initial joint positions	$[0.5, 1.5] \times$ nominal	rad
Motor strength	$[0.9, 1.1] \times$ nominal	-
Proprioception latency	$[0.005, 0.045]$	s

Table 3: **Domain Randomization Parameters.** Ranges used during the finetuning phase.

C Deployment Ablation

We conduct and ablation study on how the choice of replan margin D, caching steps m, and cache reset influence frequency, latency, and real-world locomotion performance. Performance is assessed along three criteria: forward, turning, and backward locomotion. Since aggressive caching may affect the transition between locomotion patterns, sequential omni-directional movement is specifically tested. The robot is initialized in a standing position, then commanded to move forward at $V_x = 0.8\,\text{m/s}$ for 3 s, followed by combined motion at $V_x = 0.5\,\text{m/s}$ and $V_{\text{yaw}} = 1.0\,\text{m/s}$, and finally commanded to walk backward at $V_x = -0.8\,\text{m/s}$. Latency is recorded once the computation reaches steady state, excluding rare deviations caused by resets. Results in Table 4 show that DDIM degrades motion quality compared to cached diffusion steps, and that cache resetting is crucial for robust omni-directional transitions. With these designs, diffusion planning achieves real-time operation in onboard computer while preserving motion quality.

Metrics	**A**	**B**	**C**	**D**	**E (Ours)**
Frequency (Hz)	33.1	45.8	47.7	50.1	50.0
Latency (ms)	198	77	78	22	21
Forward	✗	✗	✗	✓	✓
Turning	✗	✗	✗	✓	✓
Backward	✗	✗	✗	✗	✓

[A] D=0, m=0; [B] D=0, m=7, no refresh;
[C] DDIM (3 steps); [D] D=3, m=7, no refresh;
[E] D=3, m=7, refresh.

Table 4: **Deployment Ablation Study.** Comparing Replan Margin (D) and Cache Steps (m). Our final setting (E) achieves real-time frequency with robust omni-directional movement.

D In-the-Wild Locomotion Performance

Diffusion-MPC is evaluated on challenging real-world terrains, including soft uneven grass with varying friction and a grass slope with varying inclination, as shown in Fig. 6. The planner is deployed in a zero-shot manner without environment-specific retraining. A neural-network-based foot-lifting reward model encourages stable stepping on uneven surfaces, while a balancing reward enhances stability during traversal. For slope locomotion, regularization on the rear-calf joint position is applied adaptively: larger angles are favored for ascending slopes to prevent backward

Figure 6: **Zero-shot walking.** Left: grass. Right: grassy slope.

slipping, whereas smaller angles are encouraged for descending slopes to maintain forward stability. These results highlight that diffusion-based planning enables deployment in the wild, providing both adaptability to diverse terrains and flexible behavior modulation at test time.