Joint Model-based Model-free Diffusion for Planning with Constraints

Wonsuhk Jung*, **Utkarsh A. Mishra***, **Nadun Ranawaka Arachchige**, **Yongxin Chen**†, **Danfei Xu**†, **Shreyas Kousik**†
Georgia Institute of Technology, Atlanta, GA, USA
* indicates equal contribution, † indicates equal advising
wonsuhk.jung@gatech.edu

Abstract: Model-free diffusion planners have shown great promise for robot motion planning, but practical robotic systems often require combining them with model-based optimization modules to enforce constraints, such as safety. Naïvely integrating these modules presents compatibility challenges when diffusion's multi-modal outputs behave adversarially to optimization-based modules. To address this, we introduce Joint Model-based Model-free Diffusion (JM2D), a novel generative modeling framework. JM2D formulates module integration as a joint sampling problem to maximize compatibility via an interaction potential, without additional training. Using importance sampling, JM2D guides modules outputs based only on evaluations of the interaction potential, thus handling non-differentiable objectives commonly arising from non-convex optimization modules. We evaluate JM2D via application to aligning diffusion planners with safety modules on offline RL and robot manipulation. JM2D significantly improves task performance compared to conventional safety filters without sacrificing safety. Further, we show that conditional generation is a special case of JM2D and elucidate key design choices by comparing with SOTA gradient-based and projection-based diffusion planners. More details at: https://jm2d-corl25.github.io/

Keywords: Diffusion Models, Safety, Constrained Generation

1 Introduction

Diffusion models have advanced generative modeling of diverse, high-quality samples in challenging domains, such as images, audio, and sequential data [1, 2]. Encouraged by these successes, robotics research has increasingly leveraged diffusion for planning, control, and imitation learning [3, 4, 5, 6], exploiting its ability to capture multimodal trajectories in a model-free way.

Nevertheless, diffusion alone often falls short when robots need strict stability and safety guarantees–constraints more amenable to model-based optimization modules, such as trajectory optimizers, feedback controllers, and safety filters. Integrating model-free diffusion with model-based optimization is challenging, because these modules often struggle with alignment. Here, *alignment* for modules sharing an output space (e.g., robot trajectories) means minimizing the difference between their outputs. Alignment is particularly difficult for optimization-based safety-filters [7, 8, 9, 10], where diffusion to complete tasks can produce unsafe trajectories, while enforcing safety can degrade performance by negating generative diversity.

Alignment typically follows two strategies. The *sequential* approach minimally corrects diffusion outputs via model-based postprocessing [7, 8, 10, 11, 12], often causing module contradiction or impedance. Alternatively, *embedding* incorporates constraints into diffusion denoising [5, 13, 14, 15, 16], but can sacrifice hard guarantees or apply only to simple cases. Both strategies suffer from being *one-directional* (lacking reciprocal information sharing) and relying solely on diffusion for task

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

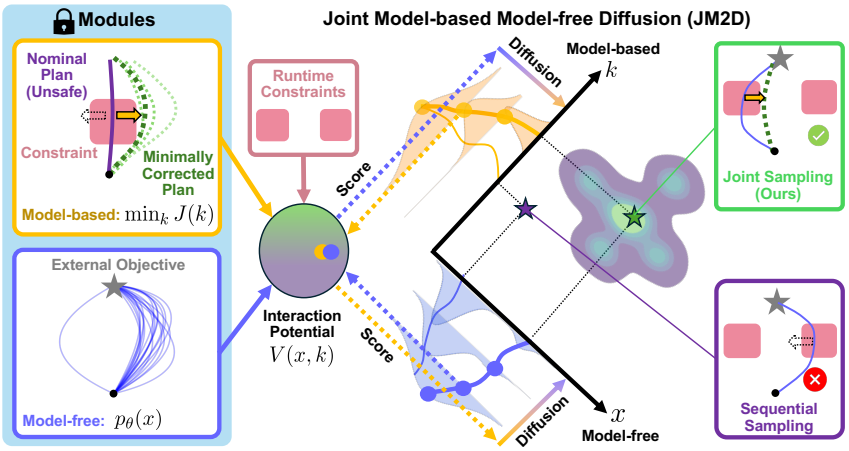


Figure 1: **Framework Overview.** We consider robotic systems composed of a model-free diffusion planner and a model-based optimization module. In this example, the diffusion policy is trained on right-skewed goal-reaching behaviors, while the model-based module can only correct plans to the right. The key challenge is to jointly sample a plan and its correction such that both modules are aware of each other's capabilities. We propose Joint Model-based Model-free Diffusion (JM2D), which defines an interaction potential combining the diffusion model, optimization module, and constraints. Starting from joint noise, our method denoises both components together, yielding mutually compatible samples (green box). In contrast, conventional sequential sampling fails to account for the optimization module's limitations, resulting in misaligned outputs (purple box).

objectives, such as task completion. Thus, we address the key challenge of how to simultaneously align model-free diffusion, model-based optimization, *and* an external objective, achieving what we call *mutual compatibility*.

To address the mutual compatibility challenge, our key insight is to first pose it as a *Joint Model-Free Model-Based Generation* (JM2G) problem of explicitly sampling simultaneously from model-free and model-based parameter spaces. To solve the JM2G problem, we take advantage of diffusion models as a generative framework and develop a *Joint Model-based Model-free Diffusion* (JM2D). To enable JM2D, we construct a joint distribution over the diffusion prior and the optimization prior, each importance weighted by an *interaction potential* that quantifies mutual compatibility, even in the absence of paired training data. As shown in Fig. 1, JM2D enables simultaneously diffusing over model-based and model-free modules while aligning both with each other and an external objective. Altogether, JM2D resolves the core mutual compatibility challenge by yielding trajectories that satisfy arbitrary, possibly non-differentiable, safety constraints by construction while preserving generative diversity.

Contributions and Paper Organization. We pose the JM2G problem to enable mutual compatibility between model-free diffusion modules, model-based optimization modules, and external objectives, introducing a novel interaction potential to relate these components (Sec. 3). To solve this, we propose the JM2D framework (Sec. 4). JM2D is based on a theoretical joint score function for simultaneous diffusion; critically, we derive a Monte Carlo score approximation that enables practical implementation even when model-based gradients are unavailable (a common scenario) and empirically improves diffusion output quality. Finally, we provide extensive experimental evaluations on offline RL benchmarks and robotic manipulation tasks (Sec. 5), demonstrating that, compared to SOTA methods with sequential sampling alignment, our joint sampling method achieves significantly improved task performance without sacrificing the safety constraint.

2 Related Work

Diffusion models and motion planning. Motion planning seeks trajectories that drive a robot from a start to a goal state while respecting kinodynamic and environmental constraints. Traditional planners—such as gradient-based optimizers [17] and sampling-based methods [18]—perform well in

structured settings but falter in highly nonconvex spaces or under discontinuous dynamics. Recently, diffusion models [19, 1, 2], which learn to sample from complex, multimodal distributions, have been applied to both data-driven [3, 20, 4, 21] and model-based [6, 22] motion planning. However, the stochastic sampling process complicates the satisfaction of explicit safety and feasibility constraints, limiting its deployment in safety-critical robotic domains.

Diffusion planning with inference-time constraints. Existing diffusion-based planners enforce constraints through three main paradigms. (i) *Post-hoc filtering* first generates diverse trajectories, then applies constrained optimization [12, 23] or safety filters [8, 9]—drawing on set-invariance theory [24], reachability analysis [25, 10], or predictive control [7]—but decoupled overrides can induce out-of-distribution actions [26] and efficiency losses [10]. (ii) *Soft-constraint guidance* uses classifier-based or classifier-free methods [27, 28] and training-free gradient steering [29, 5, 13, 14], yet demands differentiable losses and lacks strict feasibility guarantees. (iii) *Hard-constraint guidance* embeds convex feasibility via mirror maps or reflected SDEs [30, 31, 32] or projection-based corrections [15, 33, 34, 35], but is limited to convex domains and can be computationally prohibitive. Other efforts such as [36] integrate optimization during training under same-modality assumptions. Our approach generalizes beyond this by jointly sampling across modalities—plan and optimization output—via a compatibility-aware diffusion process that satisfies hard constraints without performance loss, addressing prior limitations.

3 Problem Statement and Preliminaries

This section introduces the general framework of *Joint Model-free Model-based Generation (JM2G)*, where we consider the problem of sampling from a joint distribution over a model-based and a model-free variable, given only their marginals and an interaction potential quantifying their compatibility. This formulation naturally arises in robotics, where subsystems such as planners, controllers, and safety filters are developed independently and only integrated at deployment: diffusion planners generate actions [3, 4, 20], and model-based optimization modules enforce constraints [25, 37, 11]. This creates a joint sampling problem requiring simultaneous generation of actions and corresponding responses. We first define the general version of the JM2G problem:

Problem 1. ***(Joint Model-free Model-based Generation)*** *Given a model-free generative distribution $p_\theta(x)$ over sample $x \in \mathcal{X} \subset \mathbb{R}^{n_x}$, a model-based prior $p(k)$ over sample $k \in \mathcal{K} \subset \mathbb{R}^{n_k}$ and an interaction potential $V(x,k) : \mathcal{X} \times \mathcal{K} :\to \mathbb{R}_{\geq 0}$ quantifying the compatibility of (x,k) pairs, we aim to efficiently sample from the joint distribution: $p(x,k) \propto p_\theta(x)p(k)V(x,k)$.*

While we learn the parameterized distribution $p_\theta(x)$ using a diffusion model, we assume $p(k)$ and $V(x,k)$ are only available at inference time. This reflects modular deployment scenarios where generative planners are pre-trained, and model-based optimization modules are introduced later.

In the context of robotics, we focus on a prevalent setting where the model-free component is a diffusion planner, and the model-based component is defined through *constrained optimization* [1]. Specifically: (a) we use a diffusion planner to learn the distribution of plan $x \equiv [x_t, \ldots, x_{t+H}]$ conditioned on the observation o, and (b) an optimization module that generates a model-based output k (e.g., low-level control, safety correction) by solving

$$k^* = \arg\min_k J(k \mid x) \quad \text{s.t.} \quad g(k \mid x) \leq 0 \tag{1}$$

where $J : \mathcal{X} \times \mathcal{K} \to \mathbb{R}$, $g : \mathcal{X} \times \mathcal{K} \to \mathbb{R}$ respectively denotes the objective and constraints. We aim to solve Prob. 1 with the interaction potential defined as $V(x,k) = \exp\left(-\lambda^{-1} J(k \mid x)\right) \cdot \mathbf{1}\left(g(k \mid x) \leq 0\right)$, where the temperature $\lambda \in \mathbb{R}^+$ controls the optimality and $\mathbf{1}(\cdot)$ is an indicator function. We provide a toy domain planning example in Fig. 2.

Diffusion Models. We consider diffusion models [1, 2, 38] as a generative framework. Given a data distribution $p(x_0)$, the forward noising process perturbs clean samples x_0 into noisy samples

[1] Our framework naturally includes the setting where the model-based module is defined via an explicit form $k = f(x)$ rather than an implicit form of optimization. We omit this extension for simplicity.

x_i: $p_{\alpha_i}(x_i \mid x_0) = \mathcal{N}(x_i; \sqrt{\alpha_i}x_0, (1-\alpha_i)I)$, following a noise schedule $\alpha_i \in (0,1]$. A reverse process denoises x_i to x_{i-1} using DDIM [38]:

$$p_\theta(x_{i-1} \mid x_i) = \mathcal{N}\left(x_{i-1}; \sqrt{\alpha_{i-1}}\left(\frac{x_i + (1-\alpha_i)s_\theta(x_i,i)}{\sqrt{\alpha_i}}\right) - \sqrt{1-\alpha_{i-1}-\sigma_i^2}\sqrt{1-\alpha_i}s_\theta(x_i,i), \sigma_i^2 I\right), \quad (2)$$

where $s_\theta(x_i,i)$ predicts the *score function* $\nabla_{x_i}\log p_i(x_i)$, the gradient of the log-density of noisy samples. The model is trained by minimizing the denoising score matching loss [1] by regressing over the Tweedie score of the forward kernel [39]: $s_{\alpha_i}(x_0 \mid x_i) = -\frac{x_i - \sqrt{\alpha_i}x_0}{1-\alpha_i}$.

In this work, we extend diffusion models to the *joint space* (x,k), aiming to approximate the joint score $\nabla_{x,k}\log p(x,k)$ when only the model-free score $\nabla_x \log p_\theta(x)$ is available.

4 Method

Naïvely chaining diffusion planners and model-based optimizers often fails due to conflicting objectives. Our key insight is a unified view: simultaneously generating mutually-compatible outputs via JM2D (Joint Model-based Model-free Diffusion). This section develops JM2D, starting with a unified joint diffusion process encoding mutual compatibility via a shared score function (Sec. 4.1). Since direct score computation is often intractable—due to non-differentiable objectives and reliance on clean samples—we derive a practical Monte Carlo approximation (Sec. 4.2). This allows handling hard, non-differentiable constraints without privileged information such as gradient or compromising the data fidelity of the pre-trained planner. Finally, we connect JM2D to conditional generation [40] and model-based diffusion [6] as special cases (Sec. 4.3).

4.1 Joint Diffusion Process

Challenge. A natural joint generation approach is sequential sampling: first drawing $x \sim p_\theta(x)$ from the model-free planner, then solving for $k \sim p(k \mid x)$ via model-based optimization. Alternatively, one could alternate these steps using Gibbs sampling [41]. However, in robotics, planner outputs and optimization are often tightly coupled, such as when a planner proposes an unsafe trajectory, leaving no feasible corrective action [8, 10, 11]. As a result, many x admit no compatible k, making $p(k \mid x)$ ill-defined and causing sample waste. More formally, when $p(k \mid x)$ is sparsely supported in $\mathcal{X} \times \mathcal{K}$, joint sampling becomes essential to efficiently explore feasible regions.

Design. To address the challenge, we introduce a joint diffusion process over the concatenated variable $y = [x,k]$. Specifically, we define a pseudo-forward diffusion:

$$p_{\alpha_i^y}(y_i \mid y_0) = \mathcal{N}\left(y_i \mid \sqrt{\alpha_i^y}y_0, (1-\alpha_i^y)I\right), \quad (3)$$

where $\alpha_i^y = [\alpha_i^x; \alpha_i^k]$ preserves the pre-trained model-free schedule α_i^x while letting α_i^k explore the optimization variable k. For sampling, starting from pure noise, we iteratively apply the reverse

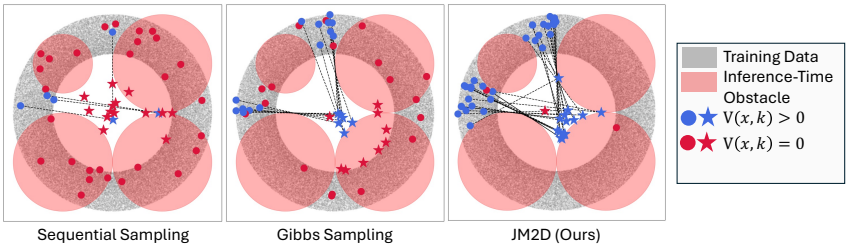


Figure 2: **Comparison of sampling methods on a toy domain.** A model-free planner $p_\theta(x)$ samples start-goal pairs in a donut-shaped region (gray), and a model-based optimization module finds the longest collision-free path connecting start and goal parameterized by k while avoiding inference-time obstacles (transparent red). We visualize 16 samples per sampling method. Sequential sampling draws x without considering k, often yielding infeasible pairs (red). Gibbs sampling alternates between $p(x \mid k)$ and $p(k \mid x)$, improving feasibility but struggling when conditionals are ill-defined. Instead, JM2D (Ours) jointly explores (x,k) via diffusion, globally searching the compatible joint sample, achieving the highest rates of mutually compatible samples (blue).

update (2) to recover a compatible joint sample. To perform this reverse process, we require the *joint score* $\nabla_{y_i} \log p_i(y_i)$, where $p_i(y_i) = \int p_{\alpha_i^y}(y_i \mid y_0) p_0(y_0) dy_0$. This score guides y_i toward higher compatibility under the interaction potential without excessively perturbing the pre-trained model-free score. Hence, our joint diffusion framework can *globally search* over y while *locally refining* according to mutual compatibility. Fig. 2 shows how joint diffusion outperforms sequential sampling in capturing compatible joint solutions.

4.2 Monte-Carlo Joint Score Approximation

Challenge. We now efficiently estimate the joint score $\nabla_{y_i} \log p_i(y_i)$ to implement the joint diffusion scheme from Sec. 4.1. This approximation is challenging in robotics for three reasons: *(a) Non-differentiability*: the interaction potentials often encode hard constraints or nonconvex optimizations, so gradients are unavailable. *(b) Clean sample dependence*: compatibility can be only evaluated on clean (fully denoised) samples, preventing naive noisy levels score composition. *(c) Data-fidelity*: the planner is pre-trained; overly aggressive corrections can destroy its learned distribution [42], so we must minimally perturb the model-free score $\nabla_{x_i} \log p(x_i)$.

Design. To address these, we derive a gradient-free Monte Carlo (MC) estimator that approximates the joint score via importance sampling, which requires only *evaluation of the interaction potential at clean joint samples* to guide the generation along the model-free reverse diffusion process.

We derive our estimator starting from

$$\nabla_{y_i} \log p_i(y_i) = \frac{\int \nabla_{y_i} \log p_{\alpha_i^y}(y_i|y_0) p_{\alpha_i^y}(y_i|y_0) p_0(y_0) dy_0}{p_i(y_i)} = \int s_{\alpha_i^y}(y_0|y_i) p_{0|i}(y_0|y_i) dy_0 \tag{4}$$

where $s_{\alpha_i^y}(y_0|y_i)$ is the Tweedie score of the forward kernel [39]. See Appx. D.1 for details. Since the posterior $p_{0|i}(y_0 \mid y_i)$ is intractable, we instead approximate it using self-normalized importance sampling with a tractable proposal $q(y_0 \mid y_i)$, giving:

$$\nabla_{y_i} \log p_i(y_i) = \frac{\int s_{\alpha_i^y}(y_0|y_i) \frac{p_{0|i}(y_0|y_i)}{q(y_0|y_i)} q(y_0|y_i) dy_0}{\int \frac{p_{0|i}(y_0|y_i)}{q(y_0|y_i)} q(y_0|y_i) dy_0} \approx \frac{\mathbb{E}_{\hat{y}_0 \sim q(\cdot|y_i)} \left[s_{\alpha_i^y}(\hat{y}_0 \mid y_i) w(\hat{y}_0) \right]}{\mathbb{E}_{\hat{y}_0 \sim q(\cdot|y_i)} \left[w(\hat{y}_0) \right]} \tag{5}$$

where $w(\hat{y}_0)$ are the importance weights. We define the proposal as

$$q(y_0 \mid y_i) = p^x_{0|i}(x_0 \mid x_i) \cdot q^k(k_0 \mid k_i), \quad q^k(k_0 \mid k_i) \triangleq \mathcal{N}\left(k_0; k_i(\alpha_i^k)^{-1/2}, \left(\tfrac{1}{\alpha_i^k} - 1\right) I\right), \tag{6}$$

where $p^x_{0|i}(x_0 \mid x_i)$ is the model-free reverse diffusion. This leads to our main theorem:

Theorem 2. ***Joint Score Approximation using Importance Sampling.*** *The joint score can be expressed as an importance-weighted combination of the scores of individually estimated clean joint samples, where the weights are determined by the interaction potential:*

$$\nabla_{y_i} \log p_i(y_i) = \frac{\mathbb{E}_{\hat{y}_0 \sim q(\cdot|y_i)} [s_{\alpha_i^y}(\hat{y}_0 \mid y_i) V(\hat{x}_0, \hat{k}_0) p_0^k(\hat{k}_0)]}{\mathbb{E}_{\hat{y}_0 \sim q(\cdot|y_i)} [V(\hat{x}_0, \hat{k}_0) p_0^k(\hat{k}_0)]} \tag{7}$$

where: $y_i = [x_i; k_i]$ *denotes the joint noisy sample,* $q(\cdot \mid y_i)$ *denotes* (5)*.*

Proof. This follows from $w(\hat{y}_0) = \frac{p_{0|i}(\hat{y}_0|y_i)}{q(\hat{y}_0|y_i)} \propto V(\hat{x}_0, \hat{k}_0) p_0^k(\hat{k}_0)$. See Appx. D.2 for the derivation. □

Having defined the proposal $q(y_0 \mid y_i)$, we estimate the joint score using MC samples and apply it in a reverse diffusion (2). See Alg. C.1 in Appx. C for details. For extensions to settings with privileged information as gradients and projection operators, see Appx. B.

Remark. In the robotics context, by encoding mutual compatibility, the interaction potential encourages sampling feasible plans, but does not guarantee feasibility ($g(k \mid x) \leq 0$ per (1)). Therefore, in practice, we postprocess infeasible joint samples (i.e., $V(x_0, k_0) = 0$) with a single model-based optimization step to guarantee feasibility. Despite this, our approach still improves performance because the sample is already aligned with the optimization (see Sec. 5).

4.3 Special Cases: Conditional Generation and Model-Based Diffusion

We conclude this section by showing that JM2D unifies Conditional Generation (CG) and Model-Based Diffusion (MBD) [6] under a common framework. This is crucial because these approaches represent widely-used paradigms in diffusion-based robotics, so unification supports direct benchmarking with CG-based methods and enables the reuse of practical heuristics from the MBD literature, such as proposal shaping and noise scheduling.

Corollary 3. *Under the interaction potential structure $V(x,k) = V^x(x)V^k(k)$ and a uniform prior $p(k)$, JM2D reduces to two independent sampling process: (1) conditional generation of x guided by $V^x(x)$, and (2) model-based diffusion over k guided by $V^k(k)$.*

Proof. Follows directly from Thm. 2. See Appx. D.3 for the proof. □

As joint sampling lacks baselines, we benchmark the CG segment against constraint-aligned planners to evaluate sampling efficiency in the absence of model-based modules (see Sec. 5.3).

5 Experiments

We experimentally validate JM2D in three ways. First, we assess the effectiveness of joint diffusion in handling complex, non-differentiable interaction potentials in a robotics safety-filtering task. Then, we illustrate how JM2D overcomes challenges in constrained generation. Finally, we assess JM2D's ability to improve conditional generation over robotics baselines.

5.1 JM2D for Safety Filtering of Robot Motion Plans

Setup. We aim to understand key performance improvements with JM2D in a safety-filtering scenario. Thus, we hypothesize that J2MD improves both safety *and* task completion over baselines. We evaluate in a 2D Maze environment (Fig. 3), where the goal is to reach the target without colliding with the walls. We use a pre-trained Diffusion Policy [3] as our model-free module and a reachability-based safety policy [37, 10] as the model-based optimization module. We introduce inference-time constraints by padding the maze walls; the diffusion planner does not see padded walls during training. We also perform an ablation to assess the tradeoff between computation time and accuracy by varying JM2D's number of Monte Carlo samples. We compare to two baselines: (a) RAIL [10], which aligns a model-free diffusion policy [3] with a reachability-based safety filter [37] via sequential sampling; and (b) a Gibbs sampler that alternates between conditional distributions over model-free and model-based components. We combine three metrics to assess mutual compatibility: **Safe Success Rate** (percentage of safe and successful trials), **Intervention Rate** (percentage of timesteps with safety filter intervention), and **Task Horizon** (duration to task completion).

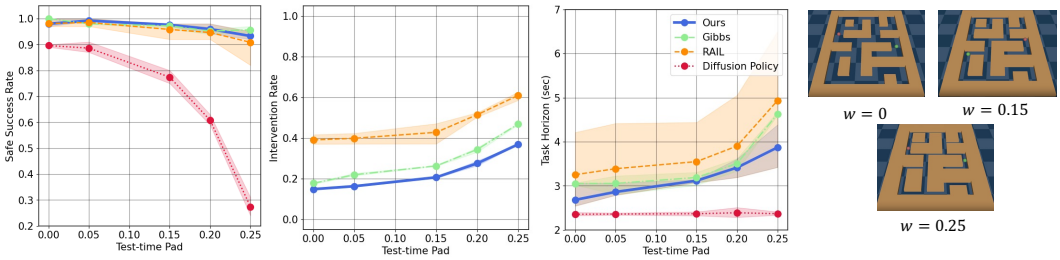


Figure 3: **Joint diffusion safety filtering increases performance without sacrificing safety.** A mobile robot (green dot) must navigate to a target (red dot) while avoiding collisions with walls. At test time, additional obstacles are introduced by padding walls with increasing thickness w. We evaluate performance across 80 trials of 5 random seeds using three metrics: *safe success rate* (left), *intervention rate* (middle), and *task horizon* (right). As wall padding increases, without the safety intervention, the baseline diffusion policy's safe success rate degrades significantly. RAIL and Gibbs sampling strategy maintains safety but exhibits increasing intervention rates. In contrast, our method consistently exhibits high safe success, with significantly fewer interventions and shorter horizons—indicating improved safety-performance trade-offs. While RAIL, Gibbs and ours all maintain strict safety guarantees, our joint sampling leads to more efficient task execution.

Result: JM2D outperforms Sequential sampling and Gibbs sampling. As shown in Fig. 3, JM2D achieves significantly lower Intervention Rate and higher Safe Success Rate over RAIL, especially as conflicts between inference-time safety constraints and the training distribution increase (e.g., with greater wall padding, the safe success rate of the diffusion policy drops sharply). This improvement results from JM2D's ability to jointly sample both the plan and its safety backup, maintaining compatibility throughout the diffusion process. RAIL's decoupled design lacks feedback, often producing incompatible samples that require external safety-filter overrides, leading to longer task horizons and reduced success. While Gibbs sampling incorporates feedback, it still suffers from higher intervention rates and longer horizons, likely due to ill-posed intermediate steps that bias guidance and reduce compatibility. None of these methods violate safety.

Result: Ablation on Monte Carlo Sample Size. As shown in Fig. 4, more samples reduce the Intervention Rate; this is due to exploring more denoising paths for x and k, enabling better mutual compatibility. However, since this increases computation time, users must balance sample size with practical efficiency.

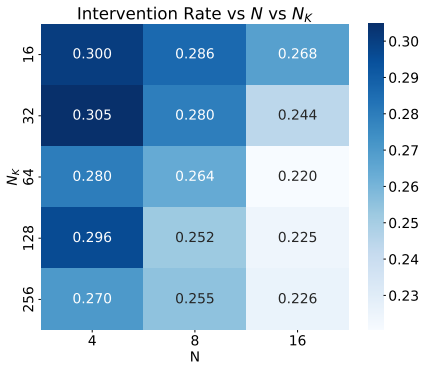
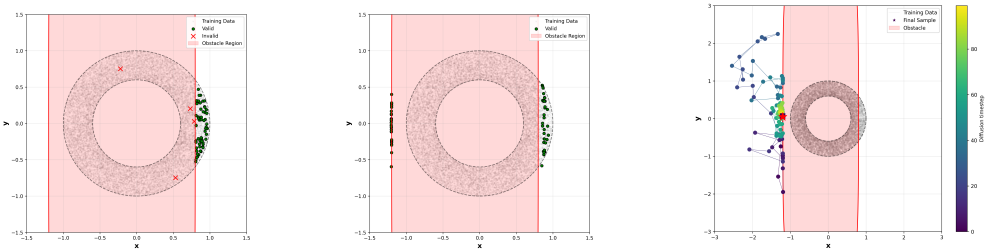


Figure 4: As the number of candidate samples (N and N_K) increases, intervention of JM2D samples by the external safety filter decreases, implying higher CA.

5.2 J2MD for Constrained Generation

Setup. We now evaluate the efficiency of our algorithm in a conditional generation setting, which we still denote as J2MD for consistency. We hypothesize that JM2D yields more valid, in-distribution samples than the baselines. We evaluate JM2D on a modified Donut Toy Domain (Fig. 2) by removing the model-based variable k and imposing tight constraints that leave only a small feasible region. The task reduces to sampling from the feasible region within the donut while staying in distribution, serving as a proxy for JM2D's ability to maintain data fidelity while complying with constraints. We compare against a projection-based conditional diffusion model that enforces feasibility by projecting noisy samples onto the constraint set at each denoising step [35]. We report two metrics: Constraint Alignment (CA), measured by the number of valid samples, and Data Fidelity, assessed by alignment with the donut distribution.

Result: JM2D Achieves Feasible, In-Distribution Samples. As shown in Fig. 5, JM2D produces many samples that are both feasible and in-distribution, despite not explicitly enforcing constraint satisfaction. In contrast, while guaranteeing feasibility, projection often generates out-of-distribution samples due to applying constraints on noisy inputs and disregarding the data distribution.



(a) Final samples (ours) (b) Final samples (projection) (c) Diffusion trajectory (projection)

Figure 5: **When does projection fail?** For constraint-guided diffusion in the `Donut` task. The gray dots show the training distribution and the red region is a test-time constraint. (a) Our method generates 60/64 valid samples within distribution. (b) Projection yields valid samples, but 31/64 lie far outside the data distribution. (c) Diffusion trajectory of a projected sample: early noise explores an out-of-distribution mode (left), and projection locks it in—despite valid, in-distribution alternatives (right), highlighting how projection can misalign constraint satisfaction with data fidelity.

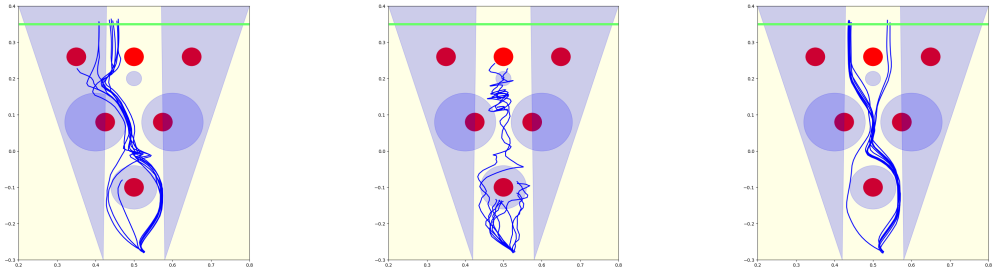
5.3 J2MD for Conditional Generation

Setup. Finally, we assess JM2D's ability to solve conditional generation and alignment tasks in robotics, which allows comparison against baselines from the literature and validates critical design choices for sampling with a non-differentiable interaction potential. We consider the Avoiding environment from D3IL [43], wherein one must generate safe trajectories for a 7DOF manipulator end effector through narrow passages under test-time constraints. We benchmark against constrained diffusion planners: SafeDiffuser [16], DPCC [15] (projection-based), and MPD [5] (gradient-based). As before, we assess *Safe Success Rate* (SSucc.), plus *Number of Constraint Violations* (#Vio) to assess conservativeness.

Result: JM2D improves alignment over projection and gradient-based diffusion planners. We find that SafeDiffuser's per-step projection induces "trap" behaviors by ignoring system dynamics [16], while trajectory-level projection methods like DPCC rely on linearized dynamics and padded uncertainty margins that overconstrain the planner, failing in cluttered, narrow-gap maneuvers (Fig. 6). Thus, early enforcement on noisy samples at higher noise levels further degrades trajectory smoothness and fidelity, leading to semantically invalid motions. Gradient-based diffusion planners like MPD [5] fare no better: by injecting cost gradients into denoising, they introduce noisy corrections that steer samples away from both constraints and the training prior, resulting in unsmooth, out-of-distribution trajectories and poor downstream performance (Table 1). JM2D instead integrates constraints into denoising, guiding samples toward in-distribution feasibility. This yields dynamically feasible trajectories that satisfy constraints and respect the learned data manifold, improving success rates and constraint violations (Table 1).

Table 1: Performance of all the algorithms in the D3IL-Avoiding task under different constraints.

Algorithm	Top Left		Top Right		Both		Cluttered	
	SSucc.	#Vio	SSucc.	#Vio	SSucc.	#Vio	SSucc.	#Vio
DPCC	**0.87**$\pm$0.04	**0.02**	0.72$\pm$0.06	0.77	0.66$\pm$0.15	1.46	0.0$\pm$0.0	6.52
SafeDiffuser	0.15 $\pm$0.12	13.18	0.07$\pm$0.02	18.27	0.01$\pm$0.01	11.66	0.01$\pm$0.01	22.92
MPD	0.13$\pm$0.13	11.97	0.08$\pm$0.06	14.85	0.03$\pm$0.03	10.80	0.02$\pm$0.02	12.71
Ours	0.77$\pm$0.10	1.13	**0.83**$\pm$0.15	**0.29**	**0.84**$\pm$0.10	**0.37**	**0.75**$\pm$0.01	**0.32**



(a) SafeDiffuser [16] (b) DPCC [15] (c) Ours

Figure 6: **Planning in Cluttered Environments.** In `Avoiding-Cluttered`, SafeDiffuser fails due to local traps; DPCC struggles to find a feasible path due to conservative obstacle padding from model uncertainty. Our method generates smooth, valid plans without privileged dynamics models.

6 Conclusion

We introduced *Joint Model-based Model-free Diffusion (JM2D)*, a unified generative framework that jointly samples from diffusion-based generative models and model-based optimization modules. This leads to a novel safety filter that improves performance while maintaining strict safety guarantees compared to traditional decoupled approaches. Our experiments show that JM2D performs robustly in realistic robotics tasks with non-differentiable constraints, outperforming conditional diffusion motion planners and conventional safety filters. Looking forward, we plan to extend JM2D to other integration problems, such as aligning generative planners with model-based tracking controllers, or aligning multiple robotics modules.

7 Limitations

Despite the improvements that we found with joint sampling, our framework still suffers from various limitations. Foremost, we achieve alignment at the cost of increased computational overhead—as we rely on denoising process to construct a Monte-Carlo sample. Automatically determining when alignment is beneficial remains an open question and is left for future work. Furthermore, our method assumes that the generative model's support overlaps the constraint-feasible region; its effectiveness fades when the learned policy becomes less multimodal. Finally, as our approach cannot strictly enforce constraints, a separate "backup" optimization module is still needed to ensure hard-constraint satisfaction.

Acknowledgments

We thank Georgia Tech AMPF for supporting this work.

References

[1] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[2] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[3] 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*, page 02783649241273668, 2023.

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

[5] J. Carvalho, A. Le, P. Kicki, D. Koert, and J. Peters. Motion planning diffusion: Learning and adapting robot motion planning with diffusion models. *arXiv preprint arXiv:2412.19948*, 2024.

[6] C. Pan, Z. Yi, G. Shi, and G. Qu. Model-based diffusion for trajectory optimization. *arXiv preprint arXiv:2407.01573*, 2024.

[7] B. Tearle, K. P. Wabersich, A. Carron, and M. N. Zeilinger. A predictive safety filter for learning-based racing control. *IEEE Robotics and Automation Letters*, 6(4):7635–7642, 2021.

[8] K.-C. Hsu, H. Hu, and J. F. Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 7, 2023.

[9] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger. Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems. *IEEE Control Systems Magazine*, 43(5):137–177, 2023.

[10] W. Jung, D. Anthony, U. A. Mishra, N. R. Arachchige, M. Bronars, D. Xu, and S. Kousik. Rail: Reachability-aided imitation learning for safe policy execution. *arXiv preprint arXiv:2409.19190*, 2024.

[11] K. P. Wabersich and M. N. Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597, 2021.

[12] T. Power, R. Soltani-Zarrin, S. Iba, and D. Berenson. Sampling constrained trajectories using composable diffusion models. In *IROS 2023 Workshop on Differentiable Probabilistic Robotics: Emerging Perspectives on Robot Learning*, 2023.

[13] K. Mizuta and K. Leung. Cobl-diffusion: Diffusion-based conditional robot planning in dynamic environments using control barrier and lyapunov functions. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13801–13808. IEEE, 2024.

[14] K. Kondo, A. Tagliabue, X. Cai, C. Tewari, O. Garcia, M. Espitia-Alvarez, and J. P. How. Cgd: Constraint-guided diffusion policies for uav trajectory planning. *arXiv preprint arXiv:2405.01758*, 2024.

[15] R. Römer, A. von Rohr, and A. P. Schoellig. Diffusion predictive control with constraints. *arXiv preprint arXiv:2412.09342*, 2024.

[16] W. Xiao, T.-H. Wang, C. Gan, and D. Rus. Safediffuser: Safe planning with diffusion probabilistic models. *arXiv preprint arXiv:2306.00148*, 2023.

[17] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE international conference on robotics and automation*, pages 489–494. IEEE, 2009.

[18] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018.

[19] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[20] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[21] C. Chen, F. Deng, K. Kawaguchi, C. Gulcehre, and S. Ahn. Simple hierarchical planning with diffusion. *arXiv preprint arXiv:2401.02644*, 2024.

[22] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing. *arXiv preprint arXiv:2409.15610*, 2024.

[23] A. Li, Z. Ding, A. B. Dieng, and R. Beeson. Constraint-aware diffusion models for trajectory optimization. *arXiv preprint arXiv:2406.00990*, 2024.

[24] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. Ieee, 2019.

[25] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[26] A. Reichlin, G. L. Marchetti, H. Yin, A. Ghadirzadeh, and D. Kragic. Back to the manifold: Recovering from out-of-distribution states. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8660–8666. IEEE, 2022.

[27] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[28] N. Botteghi, F. Califano, M. Poel, and C. Brune. Trajectory generation, control, and safety with denoising diffusion probabilistic models. *arXiv preprint arXiv:2306.15512*, 2023.

[29] H. Ye, H. Lin, J. Han, M. Xu, S. Liu, Y. Liang, J. Ma, J. Y. Zou, and S. Ermon. Tfg: Unified training-free guidance for diffusion models. *Advances in Neural Information Processing Systems*, 37:22370–22417, 2024.

[30] G.-H. Liu, T. Chen, E. Theodorou, and M. Tao. Mirror diffusion models for constrained and watermarked generation. *Advances in Neural Information Processing Systems*, 36:42898–42917, 2023.

[31] A. Lou and S. Ermon. Reflected diffusion models. In *International Conference on Machine Learning*, pages 22675–22701. PMLR, 2023.

[32] N. Fishman, L. Klarner, E. Mathieu, M. Hutchinson, and V. De Bortoli. Metropolis sampling for constrained diffusion models. *Advances in Neural Information Processing Systems*, 36: 62296–62331, 2023.

[33] J.-B. Bouvier, K. Ryu, K. Nagpal, Q. Liao, K. Sreenath, and N. Mehr. Ddat: Diffusion policies enforcing dynamically admissible robot trajectories. *arXiv preprint arXiv:2502.15043*, 2025.

[34] J. Liang, J. K. Christopher, S. Koenig, and F. Fioretto. Simultaneous multi-robot motion planning with projected diffusion models. *arXiv preprint arXiv:2502.03607*, 2025.

[35] J. K. Christopher, S. Baek, and N. Fioretto. Constrained synthesis with projected diffusion models. *Advances in Neural Information Processing Systems*, 37:89307–89333, 2025.

[36] G. Giannone, A. Srivastava, O. Winther, and F. Ahmed. Aligning optimization trajectories with diffusion models for constrained design generation. *Advances in Neural Information Processing Systems*, 36:51830–51861, 2023.

[37] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan. Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots. *The International Journal of Robotics Research*, 39(12):1419–1469, 2020.

[38] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[39] C. Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.

[40] Y. Huang, A. Ghatare, Y. Liu, Z. Hu, Q. Zhang, C. S. Sastry, S. Gururani, S. Oore, and Y. Yue. Symbolic music generation with non-differentiable rule guided diffusion. *arXiv preprint arXiv:2402.14285*, 2024.

[41] W. R. Gilks, S. Richardson, and D. Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

[42] Y. Wang, L. Wang, Y. Du, B. Sundaralingam, X. Yang, Y.-W. Chao, C. Perez-D'Arpino, D. Fox, and J. Shah. Inference-time policy steering through human interactions. *arXiv preprint arXiv:2411.16627*, 2024.

[43] X. Jia, D. Blessing, X. Jiang, M. Reuss, A. Donat, R. Lioutikov, and G. Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6pPYRXKPpw.

[44] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[45] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pages 136–145. PMLR, 2017.

[46] L. Pineda, T. Fan, M. Monge, S. Venkataraman, P. Sodhi, R. T. Chen, J. Ortiz, D. DeTone, A. Wang, S. Anderson, et al. Theseus: A library for differentiable nonlinear optimization. *Advances in Neural Information Processing Systems*, 35:3801–3818, 2022.

[47] D. P. Kingma, M. Welling, et al. Auto-encoding variational bayes, 2013.

[48] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

A Highlighted Result

A.1 Hardware

Setup We implement JM2D on a Franka robot. For our task, we chose to pick up a mug while avoiding obstacles. These obstacles were not present in the training data for the diffusion policy. To train the model, we collect 100 demonstrations of the task through teleoperation of the robot, grasping the mug in different locations like the rim and the handle. The input to the diffusion policy (DP) is the SE(3) poses of the mug (acquired through ArUco markers [44]), and the proprioceptive information of the robot. The output of the model is the absolute end-effector pose (position and translation). During testing, we placed novel obstacles in the robot's workspace as shown in Fig. A.1. The robot then has to complete the task without hitting any of the obstacles.

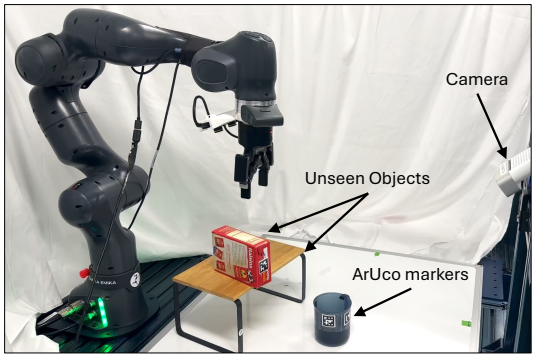


Figure A.1: Real robot experiment setup

We rollout the pre-trained vanilla DP in the cluttered scenario. Since there are obstacles in the scenario previously unseen by the DP, it collides with them and fails. This is expected as naive DP is uninformed action sampling. We show this in Fig. A.2 (top). On the other hand, we deploy DP combined with a reachability-based safety filter deployed through our JM2D sampling framework. We see an improved safety guarantee as the sampled actions respond to the backup planner and follow a safe trajectory without compromising task success as illustrated in Fig. A.2 (bottom).

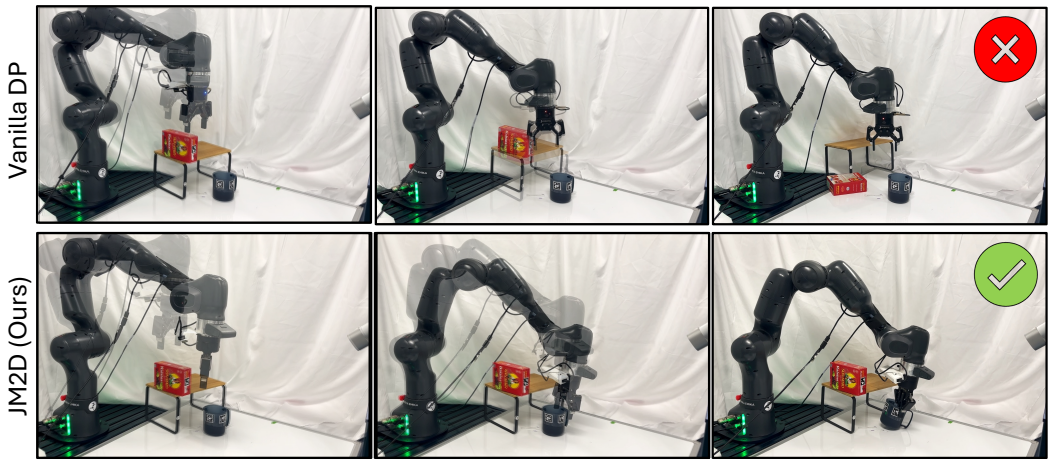


Figure A.2: (Top) Naïve DP fails to execute the task of mug-pickup when it faces obstacles unseen in the training data. (Bottom) JM2D-based composition of DP and reachability-based safety filter successfully completes the task without compromising safety.

Additionally, investigating into the diffusion process also shows the multi-modality considered by our method before choosing the final trajectory. With our JM2D framework, we can guide the diffusion sampling process to choose the safest mode. We show it in Fig. A.3.

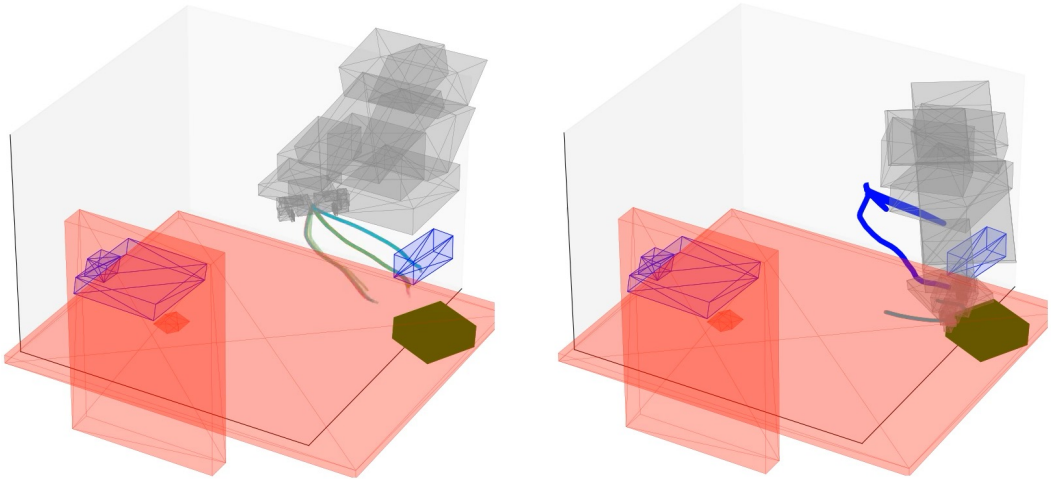


Figure A.3: (Left) Initial multi-modality in the paths that is sampled by the DP. (Right) JM2D rejects unsafe paths using safety filter implicitly throughout the denoising process to guide the generation towards safe and feasible goal reaching trajectories.

We provide the sampling hyperparameters of our real-robot setup below:

Table A.1: real-robot JM2D sampling hyperparameters

Hyperparameter	**Value**
Denoising process	`DDPM`
Number of denoising timesteps I	25
Number of action samples N	128
Number of optimization param samples N_K	128
Sample selection strategy	`Rejection Sampling`
Diffusion schedule for action α^x	`cosine`
Diffusion schedule for optimization parameter α^k	`linear`

We use the following public libraries:

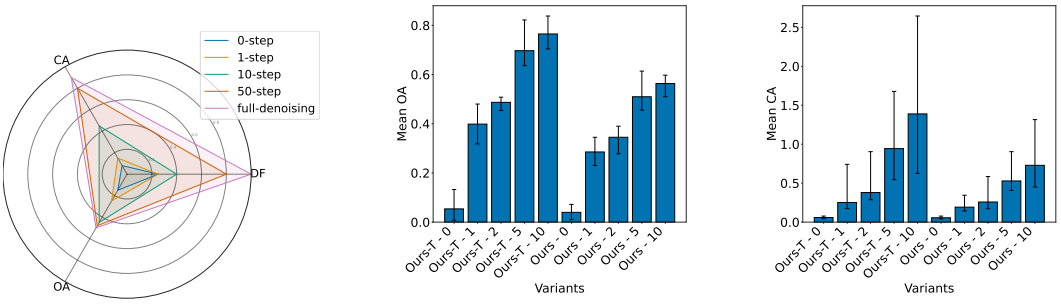
1. Diffusers: `https://huggingface.co/docs/diffusers/en/index`
2. Robomimic: `https://github.com/ARISE-Initiative/robomimic`
3. Zonopy: `https://github.com/roahmlab/zonopy`

A.2 Ablation: How does clean sample estimation impact?

Effective guidance requires accurate alignment between estimated clean samples and the true support of the target distribution. Interaction potentials in conditional generation depend on clean-sample estimates, typically obtained by either assuming noisy samples as a proxy of the clean samples or via a single-step Tweedie estimation—both of which can misalign with the data manifold. To quantify this, we introduce chamfer distance between estimated and true training samples as a measure of DF, and perform an ablation on the `Donut` task comparing four estimation schemes: (1) noisy samples, (2) single-step Tweedie, (3) u-step denoising + Tweedie, and (4) full-step denoising. In (3), at every noise level, we perform u-steps of further denoising to get the clean estimates via Tweedie and for (4) we completely denoise samples before evaluating interaction potential. On Donut, DF improves monotonically with the number of denoising steps u, reaching its maximum under full-step denoising. Crucially, increases in DF correlate with higher Objective Alignment (OA; safe-success rate) and lower Constraint Adherence (CA; violation count) (Fig. A.4(a)). In the robotics `Avoiding` domain(discussed further in Appx. E.1)—where true DF is unavailable—we observe the same trend: larger u yields greater OA and fewer CA violations (Fig. A.4(b,c)). Finally, gradient-based planners such as MPD [5] also benefit: computing cost gradients on Tweedie-corrected samples instead of raw noisy ones substantially reduces CA and boosts OA (discussed further in Appx. E.2).

Table A.2: The table shows the performance of all the variants of our algorithm with varying k-step denoising for clean sample estimation.

Algorithm	Top Left		Top Right		Both		Cluttered	
	SSucc.	#Vio	SSucc.	#Vio	SSucc.	#Vio	SSucc.	#Vio
Ours-0	0.06±0.05	16.44	0.07±0.06	18.0	0.0±0.02	12.98	0.01±0.02	24.33
Ours-1	0.33±0.12	4.21	0.23±0.12	6.15	0.23±0.12	6.76	0.48±0.25	2.90
Ours-2	0.37±0.21	3.84	0.28±0.12	4.12	0.34±0.18	5.80	0.48±0.30	1.71
Ours-5	0.51±0.15	2.46	0.46±0.23	1.91	0.61±0.18	2.10	0.63±0.18	1.10
Ours-10	0.51±0.19	2.22	0.60±0.24	0.95	0.57±0.20	1.56	0.71±0.23	0.76
Ours-Tight-0	0.13±0.15	12.92	0.06±0.07	18.34	0.0±0.02	12.98	0.01±0.02	23.19
Ours-Tight-1	0.47±0.24	3.54	0.33±0.23	5.24	0.32±0.18	5.76	0.34±0.12	1.35
Ours-Tight-2	0.50±0.30	3.14	0.45±0.17	2.80	0.50±0.16	3.48	0.39±0.15	1.11
Ours-Tight-5	0.68±0.23	1.83	0.65±0.19	1.15	0.82±0.09	0.59	0.63±0.12	0.65
Ours-Tight-10	0.70±0.22	1.6	0.81±0.19	0.41	0.84±0.07	0.38	0.73±0.09	0.49



(a) `Donut`: Metrics vs U-step denoising

(b) `Avoiding`: Mean OA vs U-step denoising

(c) `Avoiding`: Mean CA vs U-step denoising

Figure A.4: **Impact of clean sample quality on guidance effectiveness.** (a) On the `donut` task, increasing denoising steps improves Distribution Fidelity (DF), leading to higher OA and lower CA. (b, c) On the `Avoiding` task, where DF is not directly measurable, increased denoising similarly improves OA and reduces CA, confirming that better-aligned clean samples enhance guidance.

B Discussion: Practical Considerations of JM2D

In this section, we particularly want to emphasize the completeness of our framework in encompassing existing model-free sampling and model-based optimization for hard-constraint satisfaction.

Differentiable interaction potential. When the interaction potential arises from a differentiable optimization [45, 46], we can use pathwise gradient estimators [47, 29] for lower-variance score estimation.

Constraint enforcement strategy. If a projection operator onto the feasible set is available, one can project noisy samples back onto the constraint manifold at each diffusion step [35, 15, 33]. We compare these gradient-based and projection-based strategies to our proposed Monte Carlo estimator and show that while privileged methods enforce constraints more aggressively, they tend to excessively perturb the learned distribution compared to our approach empirically (as shown in Appx. F.1).

Design principle of the joint forward noise schedule. As shown via our formulation, JM2D can use different denoising schedules for model-free and model-based parameter. Our current formulation leverages `cosine` schedule for the model-free parameter. This aligns with several existing benchmarks of implementing diffusion policies (`https://github.com/real-stanford/diffusion_policy`). For the model-based parameter, prior works like MBD [6] and DIAL-MPC [22] have used `linear` schedules which can be found at `https://github.com/LeCAR-Lab/model-based-diffusion`. JM2D can incorporate the best of both model-free and model-based diffusion sampling.

Computational efficiency. JM2D is based on two main principles:

1. Increasing alignment between a predicted clean sample and the interaction potential via complete denoising at every reverse diffusion timestep leads to improved guidance.
2. Complete denoising at every timestep following a stochastic sampling process like DDPM [2] leads wide coverage of the training distribution that scales linearly with the number of sampled candidates. Specifically, it relies on the capability of the base diffusion model is capturing multiple principal modes of the training distribution.

Practical application of JM2D is computationally in-efficient. Instead of performing I denoising steps or I evaluations of the score function in standard diffusion sampling, JM2D requires $I(I+1)/2$ evaluations of the score function. Further batch size of JM2D sampling scales with the choice of N and N_K which are critical design parameters. Increasing N and N_K improves performance but increases batch size with $N \times N_K$. As future work, we consider adapting better diffusion samplers with JM2D, particularly:

1. DPM /DPM++ solvers: `https://github.com/LuChengTHU/dpm-solver`
2. DEIS: `https://github.com/qsh-zh/deis`

This will give us computational flexibility to consider higher values of N and N_K.

C Algorithms

We now elaborate the details of our Joint Model-based Model-free Diffusion (JM2D) framework.

Algorithm C.1 Joint Model-free Model-based Diffusion (JM2D)

Require: Model-free diffusion $p_\theta(x)$, Model-based prior $p_0^k(k_0)$, Interaction potential $V(x,k)$
Require: Joint forward noise schedule $\alpha_i^y = [\alpha_i^x, \alpha_i^k]$, Stochasticity σ_i

1: Initialize joint noise: $y_I = [x_I; k_I] \sim \mathcal{N}(\mathbf{0}, I)$
2: **for** $i = I$ to 1 **do**
3: Construct the Monte Carlo samples per (6):

$$\mathcal{X}_i, \mathcal{K}_i \leftarrow \textsc{SampleMC}(i, x_i, k_i, p_\theta(x), \alpha_i^y)$$

4: Estimate the joint score per (7):

$$\nabla_{y_i} \log p_i(y_i) = \begin{bmatrix} \nabla_{x_i} \log p_i(y_i) \\ \nabla_{k_i} \log p_i(y_i) \end{bmatrix} \leftarrow \frac{\sum_{(\hat{x}_0, \hat{k}_0) \in \mathcal{X}_i \times \mathcal{K}_i} \left[s_{\alpha_i^y}(\hat{y}_0 \mid y_i) V(\hat{x}_0, \hat{k}_0) p_0^k(\hat{k}_0) \right]}{\sum_{(\hat{x}_0, \hat{k}_0) \in \mathcal{X}_i \times \mathcal{K}_i} \left[V(\hat{x}_0, \hat{k}_0) p_0^k(\hat{k}_0) \right]}$$

5: Denoise the joint samples per (2) and Eq. (6) in [6]:

$$y_{i-1} \leftarrow \sqrt{\frac{\alpha_{i-1}^y}{\alpha_i^y}} \left(y_i + (1 - \alpha_i^y) \nabla_{y_i} \log p_i(y_i) \right) + \begin{bmatrix} \sqrt{1 - \alpha_{i-1}^x - \sigma_i^2} \sqrt{1 - \alpha_i^x} \nabla_{x_i} \log p_i(y_i) + \sigma_i z \\ 0 \end{bmatrix}, \quad z \sim \mathcal{N}(0, I)$$

6: **end for**
7: **return** $y_0 = [x_0; k_0]$

Algorithm C.2 Monte Carlo Sample Construction (SAMPLEMC)

Require: Denoising timestep i, Noisy model-free sample x_i, Noisy model-based sample k_i
Require: Diffusion model p_θ, Joint forward noise schedule α_i^y

1: Sample N clean model-free samples $\hat{x}_0^{(j)}$ from x_i via reverse steps:
2: **for** $\tau = i$ to 1 **do**
3: $\hat{x}_{\tau-1}^{(j)} \sim p_\theta(x_{\tau-1}^{(j)} \mid \hat{x}_\tau^{(j)})$, for $j = 1 \cdots N$
4: **end for**
5: Sample N_K clean model-based parameters:

$$\hat{k}_0^{(l)} \sim q^k(k_0 \mid k_i) = \mathcal{N}\left(k_0; k_i (\alpha_i^k)^{-1/2}, \left(\tfrac{1}{\alpha_i^k} - 1 \right) I \right), \quad l = 1 \cdots N_K$$

6: **return** $\{\hat{x}_0^{(j)}\}_{j=1}^N$, $\{\hat{k}_0^{(l)}\}_{l=1}^{N_K}$

D Additional Derivations

D.1 Derivation of Eq. (4)

We detail the derivation of (4), which expresses the joint score function prior to applying importance sampling.

$$\nabla_{y_i} \log p_i(y_i) = \frac{\nabla_{y_i} p_i(y_i)}{p_i(y_i)} \tag{8}$$

$$= \frac{\nabla_{y_i} \int p_{\alpha_i^y}(y_i|y_0) p_0(y_0)\, dy_0}{p_i(y_i)} \tag{9}$$

$$= \frac{\int \nabla_{y_i} \log p_{\alpha_i^y}(y_i|y_0)\, p_{\alpha_i^y}(y_i|y_0) p_0(y_0)\, dy_0}{p_i(y_i)} \tag{10}$$

$$= \int \nabla_{y_i} \log p_{\alpha_i^y}(y_i|y_0)\, p_{0|i}(y_0|y_i)\, dy_0 \tag{11}$$

$$= \int s_{\alpha_i^y}(y_0|y_i)\, p_{0|i}(y_0|y_i)\, dy_0 \tag{12}$$

We first marginalize the forward perturbation kernel from (8) to (9), then apply the log-derivative trick from (9) to (10). Using Bayes' rule, we obtain (11), and finally substitute the score term with the predefined Tweedie score $s_{\alpha_i^y}(y_0|y_i)$ in (12). Here, $p_{0|i}(y_0|y_i)$ is the target distribution for sampling, but it is intractable—motivating the use of importance sampling.

D.2 Proof of Thm. 2

Proof. It is sufficient to show that importance weight $\frac{p_{0|i}(y_0|y_i)}{q(y_0|y_i)}$ becomes proportional to $V(x_0,k_0)p_0^k(k_0)$ under the proposal distribution $q(y_0|y_i)$ given in (6). Since we use a self-normalized importance sampling scheme, any term independent of y_0 cancels out. Hence we show:

$$\begin{aligned}
\frac{p_{0|i}(y_0 \mid y_i)}{q(y_0 \mid y_i)} &= \frac{p_{0|i}(y_0 \mid y_i)}{p^x_{0|i}(x_0 \mid x_i) q^k(k_0 \mid k_i)} \\
&= \frac{p_{\alpha_i^y}(y_i \mid y_0) p_0(y_0)}{p_i(y_i)} \cdot \frac{p_i^x(x_i)}{p^x_{\alpha_i^x}(x_i \mid x_0) p_0^x(x_0) q^k(k_0 \mid k_i)} \\
&= \frac{p_i^x(x_i)}{p_i(y_i)} \cdot \frac{p_{\alpha_i^y}(y_i \mid y_0)}{p^x_{\alpha_i^x}(x_i \mid x_0) q^k(k_0 \mid k_i)} \cdot \frac{p_0(y_0)}{p_0^x(x_0)} \\
&= \frac{p_i^x(x_i)}{p_i(y_i)} \cdot \frac{\cancel{p^x_{\alpha_i^y}(x_i \mid x_0)} p^k_{\alpha_i^k}(k_i \mid k_0)}{\cancel{p^x_{\alpha_i^x}(x_i \mid x_0)} q^k(k_0 \mid k_i)} \cdot \frac{\cancel{p_0^x(x_0)} p_0^k(k_0) V(x_0,k_0)}{\cancel{p_0^x(x_0)}} \\
&= Z(y_i,i) \cdot p_0^k(k_0) V(x_0,k_0)
\end{aligned}$$

□

D.3 Proof of Cor. 3

Proof. From Thm. 2, we represent the joint score as

$$\nabla_{y_i} \log p_i(y_i) = \frac{\mathbb{E}_{\hat{y}_0 \sim q(\cdot|y_i)} \left[s_{\alpha_i^y}(\hat{y}_0 \mid y_i) V(\hat{x}_0, \hat{k}_0) p_0^k(\hat{k}_0) \right]}{\mathbb{E}_{\hat{y}_0 \sim q(\cdot|y_i)} \left[V(\hat{x}_0, \hat{k}_0) p_0^k(\hat{k}_0) \right]}.$$

Since the proposal distribution factorizes as

$$q(y_0 \mid y_i) = p^x_{0|i}(x_0 \mid x_i) \cdot q^k(k_0 \mid k_i),$$

and the interaction potential also factorizes as

$$V(x_0, k_0) = V^x(x_0) \cdot V^k(k_0),$$

we can decompose the joint score into separate expectations.

Score with respect to x_i:

$$\begin{aligned}\nabla_{x_i} \log p_i(y_i) &= \frac{\mathbb{E}_{\hat{x}_0 \sim p^x_{0|i}(\cdot|x_i)} \left[s_{\alpha_i^x}(\hat{x}_0 \mid x_i) V^x(\hat{x}_0) \right] \cdot \cancel{\mathbb{E}_{\hat{k}_0 \sim q^k(\cdot|k_i)} \left[V^k(\hat{k}_0) \right]}}{\mathbb{E}_{\hat{x}_0 \sim p^x_{0|i}(\cdot|x_i)} \left[V^x(\hat{x}_0) \right] \cdot \cancel{\mathbb{E}_{\hat{k}_0 \sim q^k(\cdot|k_i)} \left[V^k(\hat{k}_0) \right]}} \\ &= \frac{\mathbb{E}_{\hat{x}_0 \sim p^x_{0|i}(\cdot|x_i)} \left[s_{\alpha_i^x}(\hat{x}_0 \mid x_i) V^x(\hat{x}_0) \right]}{\mathbb{E}_{\hat{x}_0 \sim p^x_{0|i}(\cdot|x_i)} \left[V^x(\hat{x}_0) \right]}.\end{aligned}$$

Score with respect to k_i:

$$\begin{aligned}\nabla_{k_i} \log p_i(y_i) &= \frac{\cancel{\mathbb{E}_{\hat{x}_0 \sim p^x_{0|i}(\cdot|x_i)} \left[V^x(\hat{x}_0) \right]} \cdot \mathbb{E}_{\hat{k}_0 \sim q^k(\cdot|k_i)} \left[s_{\alpha_i^k}(\hat{k}_0 \mid k_i) V^k(\hat{k}_0) \right]}{\cancel{\mathbb{E}_{\hat{x}_0 \sim p^x_{0|i}(\cdot|x_i)} \left[V^x(\hat{x}_0) \right]} \cdot \mathbb{E}_{\hat{k}_0 \sim q^k(\cdot|k_i)} \left[V^k(\hat{k}_0) \right]} \\ &= \frac{\mathbb{E}_{\hat{k}_0 \sim q^k(\cdot|k_i)} \left[s_{\alpha_i^k}(\hat{k}_0 \mid k_i) V^k(\hat{k}_0) \right]}{\mathbb{E}_{\hat{k}_0 \sim q^k(\cdot|k_i)} \left[V^k(\hat{k}_0) \right]}.\end{aligned}$$

The score term $\nabla_{x_i} \log p_i(y_i)$ resembles the conditional guidance used in classifier-based diffusion [40]. Meanwhile, $\nabla_{k_i} \log p_i(y_i)$ corresponds to Eq. (9) in [6]. □

E Experimental Details

E.1 Task Description

Toy-Donut. This task requires the planner to generate start-goal pairs within a donut-shaped region (gray), while avoiding inference-time obstacles introduced at test time (shown in transparent red). The sample $x \in \mathbb{R}^4$ denotes the concatenated start and goal states, drawn from a model-free planner $p_\theta(x)$. The corresponding connecting trajectory is then optimized by a model-based module parameterized by $k \in \mathbb{R}^2$ to produce the longest collision-free path between the start and goal.

D4RL-PointMaze (Fig. E.5) This task requires the robot to reach a goal marked by a green dot while avoiding static obstacles. Demonstrations are drawn from the offline RL dataset, which consists mostly of trajectories that are collision-free with respect to the maze walls. The state and action spaces are $s \in \mathbb{R}^4$, representing position and velocity, and $a \in \mathbb{R}^2$, denoting 2D actuation forces.

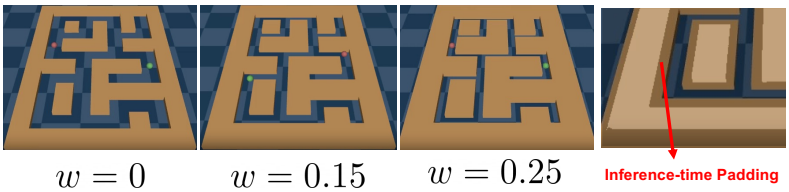
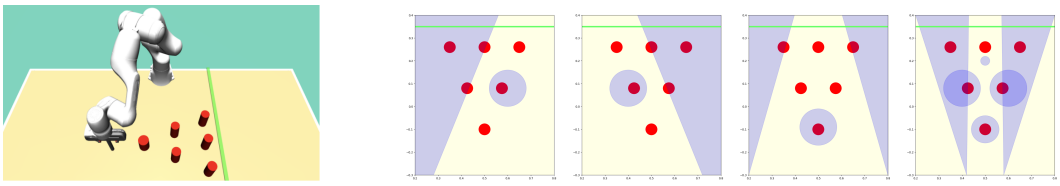


Figure E.5: **Task description of D4RL-PointMaze [48]**. (a) The robot (red) must reach the goal (green) without colliding with the wall. (b) At inference time, the wall boundaries are inflated by a width w to impose additional safety constraints. The case $w = 0$ corresponds to the original wall thickness used during training.

We train a Diffusion Policy-style planner [3] to generate sequences of actions $x = [a_t, \ldots, a_{t+H}]$ over a fixed horizon of $H = 32$, resulting in $x \in \mathbb{R}^{64}$. As the safety policy, we use RTD [37], which computes a failsafe backup action $k \in \mathbb{R}^2$ via a model-based optimization module. During evaluation, we introduce novel obstacles by inflating the wall boundaries, which were not present during training. Each policy is evaluated across 5 random seeds, with 100 simulations per seed.

D3IL-Avoiding (Fig. E.6) This task requires a manipulator to reach a goal region marked by a green line while avoiding static obstacles. Training demonstrations are all collision-free with respect to a fixed set of obstacles (shown in red), ensuring that the learned policy remains in-distribution. The state and action spaces are $s \in \mathbb{R}^4$, consisting of the current and desired end-effector positions, and $a \in \mathbb{R}^2$, denoting Cartesian end-effector velocities. The episode length is capped at 200 timesteps.



(a) Initial state of `Avoiding` task

(b) Top-down view of inference-time constraints

Figure E.6: **Task description of D3IL-Avoiding [43]**. (a) The robot must reach the green line without colliding with red obstacles. (b) At inference time, additional blue obstacles are introduced as safety constraints. The first three scenarios (Top Left, Top Right, Both Hard) are from the benchmark in DPCC [15], and the rightmost scenario (Cluttered) is newly introduced.

We train a Diffuser-style planner [4] to generate sequences of (s,a) pairs over a fixed horizon of $H = 8$, resulting in $x \in \mathbb{R}^{48}$. At test time, novel obstacles (shown in blue) not present during training are introduced, making this an instance of test-time constraint generalization. We evaluate each trained model across 5 random seeds, with 100 simulations per seed.

E.2 Baselines

DPCC [15], which enforces constraint satisfaction via projection onto a feasible set under approximate linear dynamics. Such projection-based diffusion planners enforce constraints by modifying noisy samples post hoc—typically without regard for the learned data distribution or task-specific structure embedded in it. This is problematic in settings like robotics, where feasible sets defined by test-time constraints may lie outside the support of the training data (as is often the case in Imitation Learning). In such cases, projections may find constraint-satisfying solutions that are semantically incorrect, dynamically inconsistent, or simply implausible.

SafeDiffuser [16], which employs invariance principles to ensure constraint satisfaction by design. Specifically, it applies projection at each state without modeling dynamics, leading to the well-documented "trap" behavior.

RAIL [10], which enforces the same safety filter as ours but performs filtering explicitly after sampling a diffusion plan. Such a method is highly susceptible to bad modes being sampled by the diffusion planner, such that no safe backup plans exist. Moreover, such methods lack feedback between the model-free planner and the model-based optimization module, which can result in conflicting actions and failure in completing the task objective.

MPD [5]. Motion Planning Diffusion (MPD) is a method that uses gradients to guide the denoising process. Cost functions are evaluated using noisy samples $x_i, i > 0$, and the gradient of the cost functions with respect to x_i is calculated. The sample is then stepped towards low-cost regions using weight λ before the next denoising iteration.

Algorithm E.3 Motion Planning Diffusion - Tweedie Version

Require: costs g, weights λ
Require: Diffusion forward process variances β_i, $\alpha_i := 1 - \beta_i$, $\bar{\alpha}_i := \prod_{s=1}^{i} \alpha_s$
1: $x_T \sim \mathcal{N}(\mathbf{0}, I)$
2: **for** $t = T$ to 1 **do**
3: Compute the posterior mean of x_{t-1} using model-free score:

$$\hat{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \varepsilon_\theta(x_t, t) \right)$$

4: Estimate final clean sample using Tweedie Formula:

$$\hat{x}_0 = \frac{1}{\sqrt{\alpha_{i-1}}} \left(\hat{x}_{t-1} - \sqrt{1-\bar{\alpha}_{i-1}} \hat{\varepsilon}_\theta(\hat{x}_{t-1}, i-1) \right)$$

5: Compute weighted costs:

$$\mathcal{L} = \sum_i \lambda_i g_i(\hat{x}_0)$$

6: Compute gradient of costs with respect to $\hat{x}_{t-1}$ with autograd:

$$\delta = -\Delta_{\hat{x}_{t-1}}(\mathcal{L})$$

7: Calculate the next noisy sample:

$$x_{t-1} = \hat{x}_{t-1} + g = \delta + \sigma_t z_t, \quad z_t \in \mathcal{N}(\mathbf{0}, I)$$

8: **end for**
9: **return return** x_0

F Additional Experiments

F.1 JM2D results in the smooth and safe trajectory

In this section, we show that our method enables better alignment between constraints and data fidelity compared to projection-based diffusion planners. We focus on the trade-off between hard constraint satisfaction and fidelity to the data-driven priors—such as realistic dynamics or motion patterns—that are implicitly learned by the generative model. We first inspire the failure mode of the projection via the toy example. We show how this failure mode can negatively impact in terms of the data fidelity of the generated trajectory and overall system performance in robotics.

Limitation of Projection. Projection-based diffusion planners enforce constraints by modifying noisy samples post hoc—typically without regard for the learned data distribution or task-specific structure embedded in it. This is problematic in settings like robotics, where feasible sets defined by test-time constraints may lie outside the support of the training data (as is often the case in Imitation Learning). In such cases, projections may find constraint-satisfying solutions that are semantically incorrect, dynamically inconsistent, or simply implausible.

Running Example: Donut Toy Task. We illustrate this failure mode in a 2D toy task, where a test-time constraint creates two disjoint feasible regions—only one of which overlaps with the training distribution. As shown in Fig. 5, projection moves samples toward the closest feasible region, regardless of whether the sample remains consistent with the training distribution, leading to low-fidelity samples. In contrast, our method guides the generative process toward feasible, in-distribution samples, even though it fails to provide hard constraint satisfaction.

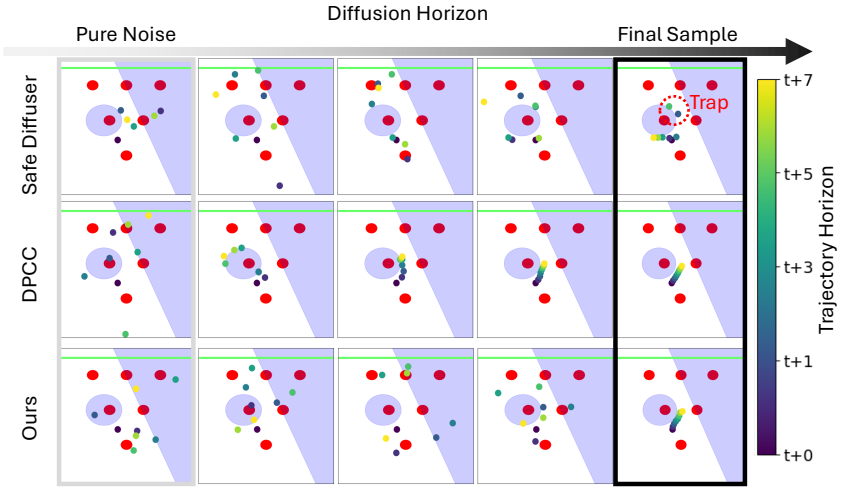
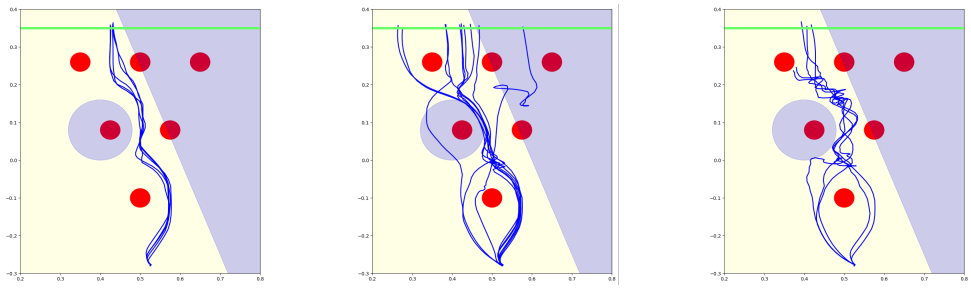


Figure F.7: **Denoising behavior in constrained trajectory generation.** Visualization of the denoising process across different methods on the `Avoiding-Left` task. Each row shows trajectory evolution from pure noise (left) to final output (right), with color indicating timestep. SafeDiffuser projects each state independently, causing trap behavior. DPCC imposes dynamics constraints at early stages, which introduces mismatches with training-time data and leads to degraded smoothness. Our method preserves randomness and structure across diffusion steps while satisfying constraints.

Effect of Incorporating Dynamics. One of the distinctive data-embedded features in demonstrations is robot dynamics. In Fig. F.7, we visualize how different methods evolve over the diffusion horizon. SafeDiffuser applies projection at each state without modeling dynamics, leading to the well-documented "trap" behavior [16]. Although trajectory-level projection, such as that in DPCC, introduces a dynamics model to avoid this issue, it does so by injecting hand-designed approximations (e.g., linearized dynamics). To account for modeling error, these methods often pad constraints

with uncertainty margins—introducing conservativeness. As seen in Fig. 6, this conservativeness can block feasible plans in cluttered environments requiring narrow-gap maneuvers.



(a) JM2D (Ours) (b) SafeDiffuser [16] (c) DPCC [15]

Figure F.8: **Comparison of trajectory smoothness and safety.** We compare sampled trajectories from different methods on the `Avoiding-Left` task. Our method achieves the best balance between data fidelity and constraint satisfaction, resulting in smoother trajectories that avoid collisions.

Mismatched Training Distribution. A subtler failure is shown in Fig. F.7, where dynamics-aware projections force early denoising steps to adhere to realistic trajectories. However, forward diffusion degrades trajectory structure, and hence noisy samples *should be* dynamically infeasible. Enforcing dynamics constraints on these corrupted inputs forces the denoiser to operate on unrealistic, out-of-distribution samples. This mismatch leads to degraded denoising quality and non-smooth outputs, as visualized in Fig. F.8. In contrast, our method maintains the generative structure across the entire diffusion trajectory, leading to smoother, higher-fidelity results.

System Performance. Finally, we argue that although projection-based diffusion motion planners enforce hard constraint satisfaction at each horizon, their tendency to produce low-fidelity trajectories leads to degraded downstream behavior. In contrast, while our method does not guarantee per-sample constraint satisfaction, it reduces overall constraint violations by naturally steering the generative process toward safe and plausible solutions. This results in improved task performance, as seen in Table 1, where our method consistently achieves higher success rates and fewer constraint violations compared to projection-based baselines.

F.2 JM2D is robust to the backup planner design quality

Table F.3: Robustness of JM2D to degraded backup–planner capabilities.

Algorithm	Original			$x+y+$			$x-y-$		
	SSucc.	SHor.	Int%	SSucc.	SHor.	Int%	SSucc.	SHor.	Int%
RAIL	0.90	351.7	0.52	0.62	417.3	0.61	0.60	367.7	0.72
JM2D (Ours)	**0.94**	**297.7**	**0.24**	**0.76**	**259.7**	**0.40**	**0.88**	**366.3**	**0.47**

Robustness to Backup–Planner Quality. Safety–filter frameworks must tolerate imperfect backup policies in real robots, so we evaluate JM2D under three variants: (1) *Original*, which performs an accelerate–brake maneuver (accelerate for 0.5 s, then decelerate to rest); (2) $x+y+$, which allows acceleration only in the positive x and y directions; and (3) $x-y-$, which allows acceleration only in the negative x and y directions. For each variant we execute 50 roll-outs with inference-time wall padding $w = 0.20$ and record the *Safe Success Rate* (SSucc.), *Successful Horizon* (SHor.), and *Intervention Ratio* (Int %), reported in Table F.3. Constraining the backup planner sharply degrades RAIL: safe success falls from 0.90 to 0.60 while interventions climb from 0.52 to 0.72. In contrast, JM2D remains resilient (SSucc. ≥ 0.76 across all variants) and roughly halves the intervention frequency, suggesting that its diffusion planner anticipates backup-planner limitations and produces plans that remain compatible even with weakened safety capabilities.