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

Safe Model Predictive Diffusion with Shielding

Taekyung Kim[1], Keyvan Majd[2], Hideki Okamoto[2], Bardh Hoxha[2], Dimitra Panagou[1,3], Georgios Fainekos[2]

***Abstract*— Generating safe, kinodynamically feasible, and optimal trajectories for complex robotic systems is a central challenge in robotics. This paper presents Safe Model Predictive Diffusion (Safe MPD), a training-free diffusion planner that unifies a model-based diffusion framework with a safety shield to generate trajectories that are both kinodynamically feasible and safe by construction. By enforcing feasibility and safety on all samples during the denoising process, our method avoids the common pitfalls of post-processing corrections, such as computational intractability and loss of feasibility. We validate our approach on challenging non-convex planning problems, including kinematic and acceleration-controlled tractor-trailer systems. The results show that it substantially outperforms existing safety strategies in success rate and safety, while achieving sub-second computation times. [Project Page][1] [Code] [Video]**

I. INTRODUCTION

Trajectory optimization is a cornerstone of robotics, enabling autonomous systems to generate goal-oriented motions consistent with their dynamics. Yet traditional nonlinear programming often struggles with the challenges inherent in real-world robotics tasks, such as non-convex objectives, complex nonlinear dynamics, and high-dimensional state-control spaces. Motivated by these challenges, diffusion-based planners have emerged as a compelling paradigm for trajectory optimization, viewing planning as probabilistic inference over trajectories and generating low-cost solutions by progressively denoising samples [1], [2].

Model-Based Diffusion (MBD) [3] strengthens this paradigm by replacing learned score networks with principled scores derived from the system dynamics and cost function. This *training-free* approach avoids collecting large datasets of expert demonstrations and naturally supports *test-time* adaptation to new tasks. However, applying MBD to constrained kinodynamic planning faces two fundamental issues: (i) *sampling inefficiency*, since feasibility and safety constraints concentrate probability mass onto a thin manifold, and (ii) the *absence of safety guarantees*, which is unacceptable in safety-critical applications.

A. Related Work

There have been several attempts to impose safety on trajectories generated by diffusion models, primarily in model-free contexts. A straightforward approach is **filtering**, where generated trajectories are discarded if they are deemed unsafe [4], [5]. This is generally sample-inefficient because it may reject a large number of samples, and it is especially problematic when feasible trajectories occupy only a thin manifold. Another popular technique is **guidance**, which steers the denoised trajectory towards safe regions by gradient descent [1], [6]–[8]. While effective, this *post-processing correction* can result in kinodynamically infeasible trajectories, undermining a critical requirement for many robotic systems to reliably execute planned motions. Furthermore, constructing a differentiable landscape for the safety objective is often non-trivial, particularly with non-convex obstacles or complex robot geometries. A third category of methods performs hard **projection** of the states onto the safe set [9]–[11]. This process can be computationally intensive, especially when the projection operation is non-convex and the state space is high-dimensional. Finally, some methods integrate **barrier functions** directly into the diffusion process [12], [13]. For instance, SafeDiffuser proposes solving a Quadratic Program (QP) with Control Barrier Function (CBF) constraints at each denoising step [13]. The computational overhead introduced by this method renders it intractable within the model-based diffusion framework, as it requires solving QPs for every parallel trajectory sample.

B. Contributions

In this paper, we propose a novel diffusion-based trajectory optimization algorithm, Safe Model Predictive Diffusion (Safe MPD$^{\blacktriangledown}$). The main contributions of this work are:

- We propose Safe MPD$^{\blacktriangledown}$ that integrates a safety shield directly into the diffusion process to guarantee kinodynamic feasibility and safety by construction.
- We empirically show a dramatic improvement in sample efficiency by enforcing that all sampled trajectories within the denoising process are both feasible and safe.
- We demonstrate that our approach is highly computationally efficient, achieving *sub-second* planning times through a parallelized GPU implementation of our shielding mechanism.
- We validate our method on challenging non-convex planning tasks including tractor-trailer systems, showing that it outperforms existing safety strategies, without requiring model-specific hyperparameter tuning.

II. PRELIMINARIES

A. Problem Formulation

Consider a discrete-time nonlinear system:

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{1}$$

This research was performed while Taekyung Kim was an intern at Toyota Motor North America, Research & Development.

[1]Department of Robotics, [3]Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, 48109, USA taekyung@umich.edu, dpanagou@umich.edu

[2]Toyota Motor North America, Research & Development, Ann Arbor, MI, 48105, USA <first_name.last_name>@toyota.com

[1]Project page: https://www.taekyung.me/safe-mpd

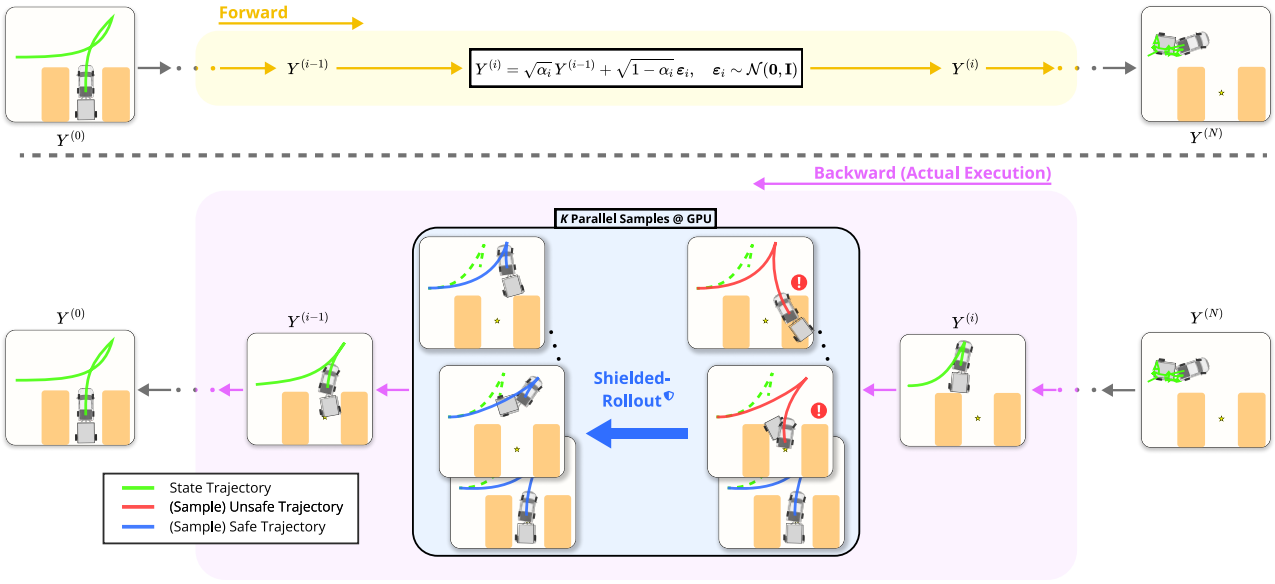

Fig. 1: Overview of the Safe Model Predictive Diffusion (Safe MPD) algorithm. (a) The forward process gradually adds noise to an optimal trajectory. (b) Reverse (denoising) process with shielded rollout: from the current noisy estimate $Y^{(i)}$, K perturbed candidates are drawn; some are initially unsafe (e.g., collisions or jackknifing for the tractor-trailer). Shielded rollout transforms each candidate into a kinodynamically feasible and safe trajectory, after which weighted averaging and score ascent update $Y^{(i-1)}$.

where $\boldsymbol{x}_t \in \mathcal{X} \subset \mathbb{R}^n$ is the state at time step $t \in \mathbb{Z}^+$, and $\boldsymbol{u}_t \in \mathcal{U} \subset \mathbb{R}^m$ is the control input at time step t, with $\mathcal{U}$ being a set of admissible controls for System (1). The function $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ represents the dynamics.

Given the initial condition $\boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}}$, we aim to solve a general trajectory optimization problem using diffusion:

$$\{\boldsymbol{x}_t^*\}_{t=1}^{T}, \{\boldsymbol{u}_t^*\}_{t=0}^{T-1} := \underset{\boldsymbol{x}_{1:T}, \boldsymbol{u}_{0:T-1}}{\arg\min} \quad J(Y) \tag{2a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t),\ t = 0, ..., T-1, \tag{2b}$$

$$g(\boldsymbol{x}_t) \leq 0,\ t = 0, ..., T, \tag{2c}$$

$$\boldsymbol{u}_t \in \mathcal{U},\ t = 0, ..., T-1. \tag{2d}$$

We denote $\tau_{\boldsymbol{x}} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T]$ as the state trajectory and $\tau_{\boldsymbol{u}} := [\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{T-1}]$ as the control sequence. We use $Y := \{\tau_{\boldsymbol{x}}, \tau_{\boldsymbol{u}}\}$ to denote all decision variables. The objective function J in (2a) is defined as $J(Y) = l_T(\boldsymbol{x}_T) + \sum_{t=0}^{T-1} l_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$, where $l_t : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ and $l_T : \mathcal{X} \to \mathbb{R}$ are the stage cost and the terminal cost. $g : \mathcal{X} \to \mathbb{R}$ represents the inequality constraint. The parameter T represents the planning horizon.

Classical approaches solve (2) via nonlinear programming, but these methods often struggle with non-convex objectives and constraints, complex nonlinear dynamics, and high-dimensional state space $\mathcal{X}$ and control space $\mathcal{U}$. Recently, an alternative paradigm that bypasses these difficulties has gained significant interest by recasting trajectory optimization as a sampling problem and directly generating samples from the optimal trajectory distribution [1]–[3], [7].

Trajectory Optimization as Probabilistic Sampling: We associate to (2) a target distribution p_0 on trajectories that factors into (i) optimality (2a), (ii) dynamical feasibility (2b), and (iii) constraint satisfaction (2c):

$$p_0(Y) \propto p_J(Y)\, p_f(Y)\, p_g(Y), \tag{3}$$

with

$$p_J(Y) \propto \exp\left(-J(Y)/\lambda\right),\ \lambda > 0, \tag{4a}$$

$$p_f(Y) \propto \prod_{t=0}^{T-1} \delta\left(\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)\right), \tag{4b}$$

$$p_g(Y) \propto \prod_{t=0}^{T} \mathbb{1}\left(g(\boldsymbol{x}_t) \leq 0\right), \tag{4c}$$

where $\delta(\cdot)$ is a Dirac delta function and $\mathbb{1}(\cdot)$ is an indicator function. In words, p_0 places probability mass on trajectories that are kinodynamically feasible and satisfying constraints, and exponentially favors low-cost ones. Obtaining the solution Y^* from solving the trajectory optimization problem in (2) is equivalent to sampling from the target distribution in (3) given a low temperature $\lambda \to 0$ [3].

In trajectory optimization tasks, since the dynamics f and objective function J are known, we can evaluate the unnormalized probability $p_0(Y)$ for any given trajectory Y. However, sampling directly from the target distribution p_0 is generally intractable because of its high dimensionality and the sparsity of the feasible manifold.

B. Model-Based Diffusion

To effectively sample from the target distribution p_0, diffusion iteratively refines samples starting from pure noise, which can be easily drawn from an isotropic Gaussian distribution. These denoising steps are referred to as the backward (or reverse) process, which reverses a predefined forward process that gradually corrupts data into pure noise [14].

Forward (noising) process: Let us denote the variance schedule as $\{\beta_i \in (0,1)\}_{i=1}^{N}$, then $\alpha_i := 1 - \beta_i$, and $\bar{\alpha}_i := \prod_{k=1}^{i} \alpha_k$ [14]. Starting from $Y^{(0)} \sim p_0(\cdot)$, the forward

Markov chain is

$$Y^{(i)} = \sqrt{\alpha_i}\, Y^{(i-1)} + \sqrt{1-\alpha_i}\, \boldsymbol{\varepsilon}_i, \quad \boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{5}$$

so that

$$p_{i|i-1}(\cdot \mid Y^{(i-1)}) \sim \mathcal{N}\left(\sqrt{\alpha_i}\, Y^{(i-1)}, (1-\alpha_i)\mathbf{I}\right). \tag{6}$$

Since the noise at each noising step is independent, it yields

$$p_{i|0}(\cdot \mid Y^{(0)}) \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_i}\, Y^{(0)}, (1-\bar{\alpha}_i)\mathbf{I}\right). \tag{7}$$

Reverse (denoising) process via Monte Carlo score ascent: The reverse process aims to recover a sample to the target distribution p_0 by starting with a sample drawn from $Y^{(N)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This involves iteratively evaluating the posterior distribution, where the marginal is given by: $p_{i-1}(Y^{(i-1)}) = \int p_{i-1|i}(Y^{(i-1)} \mid Y^{(i)}) p_i(Y^{(i)})\, \mathrm{d}\, Y^{(i)}$. Unlike standard (model-free) diffusion models that rely on a neural network learned with a large number of data to estimate the score function [14], Model-Based Diffusion (MBD) exploits the prior information of the dynamics model f and objective function J to evaluate the score directly [3].

In each denoising step from i to $i-1$, MBD performs one-step gradient ascent on $\log p_i(Y^{(i)})$:

$$Y^{(i-1)} = \frac{1}{\sqrt{\alpha_i}}\left(Y^{(i)} + (1-\bar{\alpha}_i)\nabla_{Y^{(i)}} \log p_i(Y^{(i)})\right). \tag{8}$$

The crucial insight of MBD is to estimate the score function $\nabla_{Y^{(i)}} \log p_i(Y^{(i)})$ using a Monte Carlo approximation:

$$\nabla_{Y^{(i)}} \log p_i(Y^{(i)}) \approx -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i}\overline{Y}^{(i)}. \tag{9}$$

Here, $\overline{Y}^{(i)}$ is a weighted average of a batch of candidate samples $\mathcal{Y}_k^{(i)}$, $k = 1, \ldots, K$, which are drawn from the proposal distribution

$$\mathcal{Y}_k^{(i)} \sim \mathcal{N}\left(\frac{Y^{(i)}}{\sqrt{\bar{\alpha}_i}}, \left(\frac{1}{\bar{\alpha}_i} - 1\right)\mathbf{I}\right), \tag{10}$$

and then averaged with the weights evaluated using the known target distribution p_0:

$$\overline{Y}^{(i)} := \frac{\sum_{k=1}^{K} \mathcal{Y}_k^{(i)}\, p_0(\mathcal{Y}_k^{(i)})}{\sum_{k=1}^{K} p_0(\mathcal{Y}_k^{(i)})} \tag{11}$$

In essence, at each denoising step, MBD generates a set of potential trajectories that are perturbed around the current noisy estimate, scores them using the true objective and constraints encoded in p_0, and uses this information to perform a principled update, guiding the sample from noise toward an improved solution. With the number of diffusion steps $N = 1$, MBD reduces to the Cross Entropy Method (CEM) [15].

Remark 1. *As the generation and evaluation of these K candidate samples are independent, this process is highly parallelizable. With the aid of modern GPUs/TPUs, each denoising step can be significantly accelerated [16].*

Remark 2. *MBD can be used for generic optimization problems other than trajectory optimization.*

III. SAFE MODEL PREDICTIVE DIFFUSION

Although MBD can map noise to low-cost trajectories without learned neural networks, its direct application to (2) reveals two fundamental challenges.

First, sampling efficiency can be extremely low. Recall the target distribution is given as $p_0(Y) \propto p_J(Y)\, p_f(Y)\, p_g(Y)$. The Dirac delta function for equality constraint $p_f(\cdot)$ and the indicator function for inequality constraint $p_g(\cdot)$ assign non-zero probability density only to kinodynamically feasible and constraint satisfying trajectories, which form a thin manifold of Lebesgue measure zero. Consequently, nearly all samples $\mathcal{Y}_k^{(i)}$ will receive zero weights, rendering the Monte Carlo update ineffective.

Second, while MBD inherits the computational advantages of sampling-based methods like CEM [15] and MPPI [16], it also shares their fundamental drawback: a lack of safety guarantees. Even if the candidate samples are filtered to be safe, the weighted averaging step (11) does not guarantee that the resulting updated trajectory will remain safe [17]. Also, there is no straightforward way in these frameworks to ensure that the system can be rendered safe for all future time from the terminal state $\boldsymbol{x}_T$.

In this work, we introduce ***Safe Model Predictive Diffusion (Safe MPD)*** to address these limitations: (i) we ensure that all trajectory samples at every denoising step are feasible (Sec. III-A) and safe (Sec. III-B-Sec. III-C), dramatically improving sample efficiency; and (ii) we provide formal guarantees that the final diffusion trajectory $Y^{(0)}$ is both kinodynamically feasible and safe (Sec. III-C).

A. Model Predictive Diffusion

We first show how to ensure all trajectory samples are ***kinodynamically feasible***. For each noisy candidate $\mathcal{Y}_k$ drawn from (10), we extract its control sequence $[\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{T-1}]$. This control sequence is then simulated forward from the initial state $\boldsymbol{x}_0$ using the dynamics model f, producing a new kinodynamically feasible trajectory $\mathcal{Y}_k^f$. This feasible trajectory then replaces the original sample $\mathcal{Y}_k$ in the weighted averaging step (11). This technique is analogous to shooting methods [18] and MPPI [16], and was first adapted for MBD in [3]. We refer to this framework as ***Model Predictive Diffusion (MPD)***. However, while MPD guarantees kinodynamic feasibility, it still relies on evaluating samples using $p_J(\cdot)\, p_g(\cdot)$, which remains sample-inefficient and lacks formal safety guarantees.

B. Shielded Rollout

Now, we show how to ensure all trajectory samples and the final trajectory from MPD are safe. Unlike prior methods described in Sec. I-A that rely on post-processing corrections and can be computationally expensive, we propose ***Shielded Rollout*** to ensure safety, which can be seamlessly integrated into the MPD process. Inspired by model predictive shielding [19] and gatekeeper [20], the proposed shielded rollout ensures the system remains ***safe for all future time***.

To define the shielded rollout process, we first define the backup policy π_{backup} and the corresponding sets. We denote

Algorithm 1: `Rollout`$(\boldsymbol{x}_0, \pi, T)$

Input: Initial state $\boldsymbol{x}_0$; policy π; rollout horizon T
Output: State trajectory $\tau_{\boldsymbol{x}}$

$\boldsymbol{x} \leftarrow \boldsymbol{x}_0$; $\tau_{\boldsymbol{x}}[0] \leftarrow \boldsymbol{x}_0$
for $t \leftarrow 0$ **to** $T-1$ **do**
  $\boldsymbol{x} \leftarrow f(\boldsymbol{x}, \pi(\boldsymbol{x}))$
  $\tau_{\boldsymbol{x}}[t+1] \leftarrow \boldsymbol{x}$
return $\tau_{\boldsymbol{x}}$

$\mathcal{S}$ as the set of *(instantaneously) safe* states, i.e.,

$$\mathcal{S} := \{\boldsymbol{x} \in \mathcal{X} \mid g(\boldsymbol{x}) \leq 0\}. \tag{12}$$

Definition 1 (Controlled-Invariant Set)**.** *Given a policy $\pi : \mathcal{X} \to \mathcal{U}$, a set $\mathcal{C} \subseteq \mathcal{S}$ is* controlled-invariant *under π if for all $\boldsymbol{x}_{t_0} \in \mathcal{C}$, the solution $\boldsymbol{x}_t$ of the closed-loop system $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \pi(\boldsymbol{x}_t))$ from initial condition $\boldsymbol{x}_0 = \boldsymbol{x}_{t_0}$ satisfies $\boldsymbol{x}_t \in \mathcal{C}$ for all $t \geq t_0$.*

Definition 2 (Invariance Policy)**.** *An* invariance policy *for $\mathcal{C}$ is a policy $\pi_{\text{inv}} : \mathcal{C} \to \mathcal{U}$ that renders $\mathcal{C}$ controlled-invariant.*

Definition 3 (Recovery Policy)**.** *A policy $\pi_{\text{rec}} : \mathcal{S} \to \mathcal{U}$ is called a* recovery policy *to a set $\mathcal{C} \subseteq \mathcal{S}$ if, for the closed-loop system $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \pi_{\text{rec}}(\boldsymbol{x}_t))$, the set $\mathcal{C}$ is reachable from any state in $\mathcal{S}$ within a fixed time $T_B < \infty$, i.e.,*

$$\boldsymbol{x}_{t_0} \in \mathcal{S} \implies \boldsymbol{x}_{t_0+T_B} \in \mathcal{C}. \tag{13}$$

Given an invariance policy π_{inv} and a recovery policy π_{rec} for $\mathcal{C}$, we can define the backup policy $\pi_{\text{backup}} : \mathcal{S} \to \mathcal{U}$ as

$$\pi_{\text{backup}}(\boldsymbol{x}) = \begin{cases} \pi_{\text{inv}}(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in \mathcal{C} \\ \pi_{\text{rec}}(\boldsymbol{x}), & \text{otherwise.} \end{cases} \tag{14}$$

We assume that a backup policy π_{backup} for $\mathcal{C}$ is known, which can often be designed using established methods such as simplex architectures or reachability analysis [21]. Next, we define the condition of a *valid* state trajectory, which enables the construction of a computationally efficient monitor.

Definition 4 (Valid)**.** *A state trajectory $\hat{\tau}_{\boldsymbol{x}} = [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{T_B}]$ is* valid *if the trajectory is safe w.r.t. the safe set $\mathcal{S}$ over a finite interval:*

$$\boldsymbol{x}_t \in \mathcal{S} \quad \forall t \in \{0, \ldots, T_B\}, \tag{15}$$

and the trajectory reaches $\mathcal{C}$ at T_B:

$$\boldsymbol{x}_{T_B} \in \mathcal{C}. \tag{16}$$

Remark 3. *Critically, checking whether a trajectory is valid only requires numerical forward integration of the closed-loop system over the finite interval $\{0, \ldots, T_B\}$. This is computationally lightweight and naturally parallelizable across trajectory samples.*

The proposed shielded rollout takes in a potentially unsafe nominal control sequence $\tau_{\boldsymbol{u}}$ and produces a provably-safe trajectory $\tau_{\boldsymbol{x}}^{\shield}$. Although the system starts within $\mathcal{C}$, the nominal control input (e.g., from diffusion), which is not drawn from π_{inv}, may attempt to drive the system outside the safe set $\mathcal{S}$ to achieve task objectives. To prevent this, our method acts as a safety shield. At each time step t, it first computes the prospective next state $\hat{\boldsymbol{x}}_{t+1} := f(\boldsymbol{x}_t, \boldsymbol{u}_{\text{nom},t})$ for the nominal input $\boldsymbol{u}_{\text{nom},t} \in \tau_{\boldsymbol{u}}$. It then performs a T_B-step rollout from $\hat{\boldsymbol{x}}_{t+1}$ using the backup policy π_{backup} and checks the validity of the simulated trajectory $\hat{\tau}_{\boldsymbol{x}}$ as in Definition 4. The standard rollout is shown in Algorithm 1. If valid, the nominal input $\boldsymbol{u}_{\text{nom},t}$ is accepted; otherwise, the system switches to π_{backup} for the remainder of the horizon. The procedure is outlined in detail in Algorithm 2.

Algorithm 2: `Shielded-Rollout`$^{\shield}(\boldsymbol{x}_0, \tau_{\boldsymbol{u}})$

Input: Initial state $\boldsymbol{x}_0$; nominal control sequence $\tau_{\boldsymbol{u}}$; backup policy π_{backup}
Output: Shielded state trajectory $\tau_{\boldsymbol{x}}^{\shield}$; Shielded control sequence $\tau_{\boldsymbol{u}}^{\shield}$

$\boldsymbol{x} \leftarrow \boldsymbol{x}_0$; $\tau_{\boldsymbol{x}}^{\shield}[0] \leftarrow \boldsymbol{x}_0$
for $t \leftarrow 0$ **to** $T-1$ **do**
  $\boldsymbol{u}_{\text{nom}} \leftarrow \tau_{\boldsymbol{u}}[t]$
  $\hat{\boldsymbol{x}} \leftarrow f(\boldsymbol{x}, \boldsymbol{u}_{\text{nom}})$
  $\hat{\tau}_{\boldsymbol{x}} \leftarrow$ `Rollout`$(\hat{\boldsymbol{x}}, \pi_{\text{backup}}, T_B)$
  if *$\hat{\tau}_{\boldsymbol{x}}$ is valid by Definition 4* **then**
    $\boldsymbol{x} \leftarrow \hat{\boldsymbol{x}}$; $\tau_{\boldsymbol{x}}^{\shield}[t+1] \leftarrow \hat{\boldsymbol{x}}$; $\tau_{\boldsymbol{u}}^{\shield}[t] \leftarrow \boldsymbol{u}_{\text{nom}}$
  else
    for $t' \leftarrow t$ **to** $T-1$ **do** `// fallback to backup`
      $\boldsymbol{u}_{\text{backup}} \leftarrow \pi_{\text{backup}}(\boldsymbol{x})$; $\tau_{\boldsymbol{u}}^{\shield}[t'] \leftarrow \boldsymbol{u}_{\text{backup}}$
      $\boldsymbol{x} \leftarrow f(\boldsymbol{x}, \boldsymbol{u}_{\text{backup}})$; $\tau_{\boldsymbol{x}}^{\shield}[t'+1] \leftarrow \boldsymbol{x}$
    break
return $\tau_{\boldsymbol{x}}^{\shield}$, $\tau_{\boldsymbol{u}}^{\shield}$

Assumption 1. Our analysis is based on a discrete-time formulation. We assume that the system's continuous trajectory between two consecutive safe states, $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+1}$, remains within the safe set $\mathcal{S}$.

Theorem 1 (`Shielded-Rollout`$^{\shield}$)**.** *Given any initial state $\boldsymbol{x}_0 \in \mathcal{C}$, the shielded state trajectory $\tau_{\boldsymbol{x}}^{\shield}$ generated by Algorithm 2 enables the system to remain in the safe set $\mathcal{S}$ for all future time, i.e., $t \geq 0$.*

Proof. We prove the claim by induction.

Base Case ($t = 0$): By assumption, $\boldsymbol{x}_0 \in \mathcal{C} \subseteq \mathcal{S}$.

Induction Step: Assume that $\boldsymbol{x}_t \in \mathcal{S}$ for some $t \in \{1, \ldots, T-1\}$. We now show that the subsequent state $\boldsymbol{x}_{t+1}$ also remains in $\mathcal{S}$. As described in Algorithm 2, the algorithm evaluates the nominal control input $\boldsymbol{u}_{\text{nom},t} \in \tau_{\boldsymbol{u}}$ by first computing $\hat{\boldsymbol{x}}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_{\text{nom},t})$. This leads to two cases:

i) *Case 1 (Valid nominal step):* If the simulated trajectory $\hat{\tau}_{\boldsymbol{x}}$ from `Rollout`$(\hat{\boldsymbol{x}}_{t+1}, \pi_{\text{backup}}, T_B)$ is valid, Definition 4 guarantees $\hat{\boldsymbol{x}}_{t+1} \in \mathcal{S}$ *and* π_{backup} can drive $\hat{\boldsymbol{x}}_{t+1}$ into $\mathcal{C}$ within T_B while staying in $\mathcal{S}$. Hence $\boldsymbol{x}_{t+1} := \hat{\boldsymbol{x}}_{t+1} \in \mathcal{S}$ and $\mathcal{C}$ is reachable no later than T_B.

ii) *Case 2 (Invalid nominal step):* If it was not valid, `Shielded-Rollout`$^{\shield}$ applies π_{backup}. Since the nominal control inputs $\boldsymbol{u}_{\text{nom},(\cdot)}$ in previous steps would have been applied only when the simulated trajectories were valid, we have $\boldsymbol{x}_{t+1} \in \mathcal{S}$ by Definition 4. Moreover, the system reaches $\mathcal{C}$ within T_B under π_{rec}.

Thus $\boldsymbol{x}_{t+1} \in \mathcal{S}$ in all cases, completing the induction.

Algorithm 3: Safe Model Predictive Diffusion$^{\blacklozenge}$

Input: noise schedule $\{\bar{\alpha}_i\}_{i=1}^N$; denoising steps N; number of samples K
Output: safe optimized trajectory
Initialization: draw $Y^{(N)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $i \leftarrow N$ ***to*** 1 **do**
  `// -- Denoising step -- //`
  Sampling: $\mathcal{Y}_{1:K}^{(i)} \sim \mathcal{N}\left(\frac{Y^{(i)}}{\sqrt{\bar{\alpha}_i}}, \left(\frac{1}{\bar{\alpha}_i} - 1\right)\mathbf{I}\right)$
  Shielded Rollout: $\mathcal{Y}_{1:K}^{(i)\blacklozenge} \leftarrow$ `Shielded-Rollout`$^{\blacklozenge}(\mathcal{Y}_{1:K}^{(i)})$
  Weighted sum: $\overline{Y}^{(i)} \coloneqq \frac{\sum_{k=1}^K \mathcal{Y}_k^{(i)\blacklozenge} p_0(\mathcal{Y}_k^{(i)\blacklozenge})}{\sum_{k=1}^K p_0(\mathcal{Y}_k^{(i)\blacklozenge})}$
  Score estimate: $\nabla_{Y^{(i)}} \log p_i(Y^{(i)}) \approx -\frac{Y^{(i)}}{1-\bar{\alpha}_i} + \frac{\sqrt{\bar{\alpha}_i}}{1-\bar{\alpha}_i}\overline{Y}^{(i)}$
  Score-based update:
  $Y^{(i-1)} \leftarrow \frac{1}{\sqrt{\alpha_i}}\left(Y^{(i)} + (1-\bar{\alpha}_i)\nabla_{Y^{(i)}} \log p_i(Y^{(i)})\right)$
return `Shielded-Rollout`$^{\blacklozenge}(Y^{(0)})$

Moreover, the system at $\boldsymbol{x}_T$ can enter $\mathcal{C}$ after at most time T_B. Then π_{inv} ensures $\boldsymbol{x}_t \in \mathcal{C} \subseteq \mathcal{S}$ for all future time $t \geq T + T_B$. □

C. Main Algorithm

Finally, we present our main algorithm, Safe MPD$^{\blacklozenge}$. A schematic overview is shown in Fig. 1, and as detailed in Algorithm 3, it integrates `Shielded-Rollout`$^{\blacklozenge}$ at two critical stages.

Within the diffusion process: Within each denoising step, all K candidate trajectory samples $\mathcal{Y}_{1:K}^{(i)}$ are passed through `Shielded-Rollout`$^{\blacklozenge}$ to produce $\mathcal{Y}_{1:K}^{(i)\blacklozenge}$ that lie strictly on the feasible and safe manifold. This offers two significant advantages for trajectory optimization. (i) First, since every shielded sample $\mathcal{Y}_k^{(i)\blacklozenge}$ is guaranteed to be safe and feasible, the probability terms for feasibility and safety are constant across all such samples and can thus be disregarded from the original target distribution (3). Specifically, for any two samples, $k_1, k_2 \in \{1, \ldots, K\}$:

$$p_f(\mathcal{Y}_{k_1}^{(i)\blacklozenge}) = p_f(\mathcal{Y}_{k_2}^{(i)\blacklozenge}) \text{ and } p_g(\mathcal{Y}_{k_1}^{(i)\blacklozenge}) = p_g(\mathcal{Y}_{k_2}^{(i)\blacklozenge}). \quad (17)$$

The target distribution in Safe MPD$^{\blacklozenge}$ therefore simplifies to depend solely on the optimality factor:

$$p_0(Y) \propto p_J(Y). \quad (18)$$

This *improves sample efficiency*, since no computational effort is wasted on samples that would otherwise receive zero weight due to constraint violations. (ii) Second, the diffusion process complements the conservative nature of safety filters [11], [20]. Applying safety filters only at the last layer of the trajectory generation often overly constrains the solution, reducing overall performance. The iterative score ascent continuously pushes the trajectory distribution toward lower-cost regions, which we find empirically helps the optimizer overcome local minima while shielding preserves safety.

On the final trajectory: Furthermore, we apply `Shielded-Rollout`$^{\blacklozenge}$ to the final trajectory $Y^{(0)}$ from the diffusion process. This guarantees that the trajectory returned by Safe MPD$^{\blacklozenge}$ is kinodynamically feasible and safe, satisfying all inequality constraints by construction during its execution. Moreover, it guarantees that the system can be rendered safe from the terminal state $\boldsymbol{x}_T$ for all future time.

IV. RESULTS

Our experimental evaluations aim to answer the following key questions: **(Q1)** Can our method solve complex, non-convex trajectory optimization problems with kinodynamic constraints, where safety depends on factors like inertia and acceleration limits? **(Q2)** Is our method scalable to different dynamical systems without requiring model-specific hyper-parameter tuning? **(Q3)** Can the proposed shielded rollout be integrated into the MPD framework in a computationally efficient manner? **(Q4)** Are the resulting trajectories kinodynamically feasible and executable by tracking controllers?

A. Dynamical Systems and Environments

To address **Q1** and **Q2**, we evaluate our algorithm on a series of increasingly challenging dynamical models. The problems we consider involve scenarios where safety cannot be naïvely achieved by simply stopping due to the dynamics of high-order systems and *input constraints*. The evaluation task is an automated parking scenario, where the vehicle must navigate a cluttered environment with N_{obs} obstacles modeled as rectangles and circles to reach a target configuration. We denote the set of obstacles as $\mathcal{O} \coloneqq \bigcup_{j=1}^{N_{\text{obs}}} \mathcal{O}_j$.

We consider: (i) a kinematic bicycle, (ii) a kinematic tractor-trailer system, and (iii) an acceleration-controlled tractor-trailer system. The dynamics of the discrete-time kinematic tractor-trailer model with sampling time $T_s > 0$ is given by:

$$p_{t+1}^x = p_t^x + T_s v_t \cos(\theta_t^1),\ p_{t+1}^y = p_t^y + T_s v_t \sin(\theta_t^1), \quad (19)$$

$$\theta_{t+1}^1 = \theta_t^1 + T_s \frac{v_t}{\ell_1} \tan(\delta_t), \quad (20)$$

$$\theta_{t+1}^2 = \theta_t^2 + T_s \frac{v_t}{\ell_2}\left(\sin(\theta_t^1 - \theta_t^2) - \frac{\ell_h}{\ell_1}\cos(\theta_t^1 - \theta_t^2)\tan(\delta_t)\right). \quad (21)$$

The state of the system is $\boldsymbol{x}_{\text{tt}} = [p^x, p^y, \theta^1, \theta^2]^\top$, where (p^x, p^y) denotes the position of the rear axle of the tractor, and θ^1 and θ^2 are the heading angles of the tractor and the trailer, respectively. The control input is $\boldsymbol{u}_{\text{tt}} = [v, \delta]^\top$, where v is the longitudinal velocity of the tractor and δ is the steering angle of the front wheels. The geometric parameters ℓ_1, ℓ_2, ℓ_h denote the tractor wheelbase, trailer length, and hitch length, respectively. The kinematic bicycle model is described simply by (19)-(20).

We also introduce an acceleration-controlled tractor-trailer system with second-order dynamics (see Fig. 2). We augment the state with velocity and steering angle, yielding $\boldsymbol{x}_{\text{acc-tt}} = [p^x, p^y, \theta^1, \theta^2, v, \delta]^\top$. The control inputs are now the longitudinal acceleration and the steering rate, $\boldsymbol{u}_{\text{acc-tt}} = [a, \omega]^\top$. The dynamics is described by (19)-(21), and the following:

$$v_{t+1} = v_t + T_s\, a_t, \quad \delta_{t+1} = \delta_t + T_s\, \omega_t \quad (22)$$

In all three models, the *control inputs are* ***bounded*** ($\mathcal{U} \neq \mathbb{R}^2$).

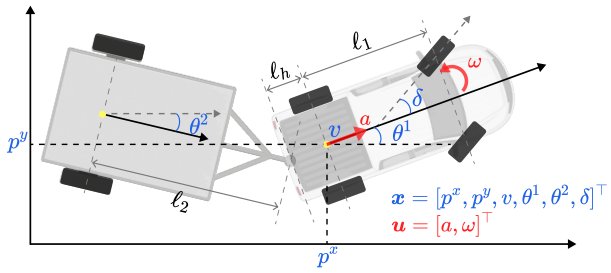

Fig. 2: Illustration of an acceleration-controlled tractor-trailer system.

Trajectory planning and control for tractor-trailer systems are challenging, particularly during backward maneuvers, due to their highly nonlinear and unstable dynamics. An additional safety constraint on the hitch angle is also required to prevent jackknifing, i.e., $|\theta^1 - \theta^2| \leq \theta_{\text{max}}$. Furthermore, the vehicle's body is composed of two disjoint geometries, the tractor $\mathcal{R}_{\text{tr}}(\boldsymbol{x}) \subset \mathbb{R}^2$ and the trailer $\mathcal{R}_{\text{tl}}(\boldsymbol{x}) \subset \mathbb{R}^2$ at $\boldsymbol{x}$, which makes the collision-free state space non-convex. This non-convexity makes safety enforcement via state projection computationally expensive.

For our shielded rollout algorithm, we design intuitive backup policies. For the kinematic bicycle and tractor-trailer models, where velocity is a control input, both π_{inv} and π_{rec} simply apply zero velocity, $v = 0$, to stop the system. For the acceleration-controlled tractor-trailer, π_{rec} applies maximum acceleration a_{max} or deceleration $-a_{\text{max}}$ to drive the velocity v to zero. Once $v = 0$ is achieved, π_{inv} applies $a = 0$ to maintain a stationary state. Further examples of backup-controller design for other dynamical systems can be found in [19], [20].

B. Experimental Setup & Baseline Methods

The (instantaneously) safe set $\mathcal{S}$ from (12) for tractor-trailers is formally defined as:

$$\mathcal{S} = \Big\{ \boldsymbol{x} \in \mathcal{X} \;\Big|\; |\theta^1 - \theta^2| \leq \theta_{\text{max}} \quad \text{(no jackknifing)}, \\ (\mathcal{R}_{\text{tr}}(\boldsymbol{x}) \cup \mathcal{R}_{\text{tl}}(\boldsymbol{x})) \cap \mathcal{O} = \emptyset \quad \text{(no collision)} \Big\}. \tag{23}$$

For the kinematic bicycle model, $\mathcal{S}$ reduces to collision avoidance with obstacles only.

We compare our proposed method against the following safety strategies commonly used for diffusion planners:

(i) Naïve Penalty: This method adds a high penalty term to the cost function J (4a) for any state that lies outside the safe set $\mathcal{S}$.

(ii) Projection: It defines a projection operator $\mathcal{P}_{\mathcal{S}}$ onto $\mathcal{S}$ subject to system dynamics:

$$\boldsymbol{u}_t^\star = \mathcal{P}_{\mathcal{S}}(\boldsymbol{u}_{\text{nom},t}; \boldsymbol{x}_t) \coloneqq \underset{\boldsymbol{u}_t \in \mathcal{U}}{\arg\min} \; \|\boldsymbol{u}_t - \boldsymbol{u}_{\text{nom},t}\|_2^2 \\ \text{s.t.} \quad \hat{\boldsymbol{x}}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t), \quad \hat{\boldsymbol{x}}_{t+1} \in \mathcal{S}. \tag{24}$$

This projection (24) is applied recursively to the K trajectory samples $\mathcal{Y}_{1:K}$ during the diffusion process, and to the final output $Y^{(0)}$, instead of shielded rollout in Algorithm 3. Projection for diffusion models was introduced in [9] and extended to trajectory planning in [11] under the assumption that the safe set is convex and the dynamics is linear. We will show that its computational overhead becomes intractable for the non-convex and nonlinear tasks in Sec. IV-C.

(iii) Guidance: This method performs gradient descent on a given state trajectory $\tau_{\boldsymbol{x}}^1 \coloneqq \tau_{\boldsymbol{x}}$ to gradually steer it away from the unsafe set:

$$\tau_{\boldsymbol{x}}^{j+1} = \tau_{\boldsymbol{x}}^j - \texttt{clip}\left(\alpha_{\text{g}} \nabla \mathcal{J}\left(\tau_{\boldsymbol{x}}^j\right), -\epsilon, \epsilon\right), \; j = 1, \ldots, N_{\text{iter}}, \tag{25}$$

where $\mathcal{J}(\cdot)$ indicates the amount of safety violation:

$$\mathcal{J}(\tau_{\boldsymbol{x}}) \coloneqq \sum_{t=1}^{T} \Bigg[\underbrace{\max\left(0, \; |\theta_t^1 - \theta_t^2| - \theta_{\text{max}}\right)}_{\text{(1) hitch-angle violation}} \\ + \underbrace{\sum_{j=1}^{N_{\text{obs}}} \max\left(0, R_j - \text{dist}\big(\mathcal{R}_{\text{tr}}(\boldsymbol{x}_t) \cup \mathcal{R}_{\text{tl}}(\boldsymbol{x}_t), \mathcal{O}_j\big)\right)}_{\text{(2) collision violation}} \Bigg]. \tag{26}$$

The number of guidance steps N_{iter} and the step size α_{g} are set to 3 and 0.05, respectively. The guidance update is clipped to a maximum magnitude ϵ [6]. R_j is the safety margin for j-th obstacle.[2] As with projection, guidance is applied to the K samples during diffusion and to $Y^{(0)}$, replacing shielded rollout in Algorithm 3.

All baselines are implemented within the MPD framework with $N = 100$, $K = 20,000$, $T = 50$, and $T_s = 0.25$ s. The running and terminal costs penalize position error and heading error relative to the goal pose. Heading errors are wrapped to $[-\pi, \pi]$, so both forward and backward parking minimize the heading objective. For tractor-trailers, the positional cost uses the minimum of the tractor and trailer positional errors to the goal, allowing the cost to be minimized regardless of whether the vehicle parks in a forward or backward orientation.

All algorithms are implemented in Python using the JAX library [22] to enable GPU-accelerated rollouts. For **Q3**, we measure the total computation time from the initial noise $Y^{(N)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to the final optimized trajectory. All experiments were conducted on an NVIDIA RTX 4090 GPU.

C. Experimental Results

The automated parking environment contains 36 obstacles. The goal position is set to the center of the designated parking slot, with the goal heading perpendicular to the slot's width. Initial states are uniformly sampled from the free space excluding trivial initial poses from which a straight maneuver could solve the task. Each method is evaluated over 100 randomized trials for each model. For hyperparameters, we tune the temperature λ (4a) and running/terminal cost weights on the kinematic bicycle model using Optuna [23]. For **Q2**, we then *reuse* these hyperparameters for both tractor-trailer systems. We report three metrics: (i) **Success Rate**: the percentage of trials where the tractor or trailer footprint enters the goal area without any safety violation; (ii) **Safety**

[2]Constructing a differentiable signed-distance function for our non-convex, articulated geometry requires costly computation, so we employ this more tractable objective.

TABLE I: Performance comparison of model-based diffusion planners with different safety strategies across three dynamics models: (A) kinematic bicycle (Bicycle), (B) kinematic tractor trailer (TT), and (C) acceleration-controlled tractor trailer (Accel. TT). For each model, 100 trials were conducted with different initial conditions. The best and comparable results are marked in **bold**. *For the projection method, the environment was simplified to contain only 6 circular obstacles near the goal position, whereas other methods were evaluated with 36 obstacles.

Metric	Model	MPD w/ Naïve Penalty	MPD w/ Projection*	MPD w/ Guidance	Safe MPD$^{\blacklozenge}$ (Ours)
Success Rate	(A) Bicycle	**100%**	**100%**	89%	**100%**
	(B) TT	64%	N/A	51%	**100%**
	(C) Accel. TT	81%	N/A	80%	**98%**
Safety Violations	(A) Bicycle	**0%**	**0%**	4%	**0%**
	(B) TT	36%	N/A	43%	**0%**
	(C) Accel. TT	19%	N/A	20%	**0%**
Computation Time	(A) Bicycle	**0.327 ± 0.009 s**	1959.816 ± 360.710 s	0.568 ± 0.016 s	**0.315 ± 0.010 s**
	(B) TT	**0.554 ± 0.027 s**	Time Out	0.998 ± 0.029 s	**0.579 ± 0.023 s**
	(C) Accel. TT	**0.575 ± 0.025 s**	Time Out	0.991 ± 0.028 s	1.631 ± 0.024 s
Kinodynamically Feasible		✓	✓	✗	✓

Violations: the percentage of trials where any constraint (collision or jackknifing) is violated; (iii) **Computation Time**. Further implementation details are available in our code repository.

The quantitative results are summarized in Table I. On the kinematic bicycle model, all methods except guidance achieve a 100% success rate, since this task is comparatively easy and does not involve a trailer. Guidance exhibits safety violations across all systems because it provides no formal safety guarantee. Moreover, because it applies a post-processing correction to the state trajectory, the resulting trajectory may no longer be kinodynamically feasible. For the projection method only, we simplify the environment to contain just 6 circular obstacles near the goal position. Even then, it requires an average of 32.664 minutes for the kinematic bicycle model, although it guarantees safety. For the tractor-trailer systems, it hits the 1-hour timeout in all trials.

When tested on the more complex tractor-trailer systems, the limitations of the baselines become evident. Both the Naïve Penalty and Guidance methods show a significant drop in success rate and a sharp increase in safety violations, highlighting their inability to handle the challenges of higher-dimensional, non-convex problems. In contrast, our Safe MPD$^{\blacklozenge}$ maintains a perfect 0% safety violation rate across all models and tasks **(Q1)**. It also achieves near-perfect success rates of 100% and 98% for the kinematic and acceleration-controlled tractor-trailer models, respectively (see Fig. 3(a)), demonstrating its scalability even without model-specific hyperparameter tuning **(Q2)**. Fig. 3(b) shows a successful trajectory that includes multi-point turns in a tight space, with no hitch-angle violations or collisions. Regarding **Q3**, our method's computational performance is highly competitive. The primary overhead of the shielded rollout comes from the finite-horizon rollouts under the backup policy during validity checks, which are highly parallelizable and can be computed efficiently on a GPU. As a result, Safe MPD$^{\blacklozenge}$ achieves computation times comparable to the fastest (but unsafe) baseline on the kinematic models. The higher computation time for the acceleration-controlled model stems from its second-order dynamics. Verifying safety requires a longer horizon T_B with the backup policy π_{backup} to ensure the system can be brought to a full stop within $\mathcal{C}$. This sensitivity correlates with tighter input bounds: stricter input limits require more recovery steps to verify validity.

Integration test with a tractor-trailer navigation stack. For **Q4**, we integrate our algorithm into our existing tractor-trailer navigation framework [24], replacing the Hybrid A* planner with Safe MPD$^{\blacklozenge}$. The time to generate a feasible path drops from several minutes to under a second. Because our method generates kinodynamically feasible trajectories, the framework's downstream tracking controller (BR-MPPI [25]) reliably tracks the resulting diffusion trajectories with multi-point turns (see our project page).

V. CONCLUSION

In this work, we introduced Safe Model Predictive Diffusion (Safe MPD$^{\blacklozenge}$), a novel framework for trajectory optimization that integrates a formal safety shield directly into the denoising process of a model-based diffusion planner. Our method demonstrates three advantages that are critical for real-world robotics: the generated trajectories are (i) *kinodynamically feasible by construction*, (ii) *provably safe*, and (iii) *computationally efficient* through batched rollouts and Monte Carlo score ascent on a GPU. The strong performance and sub-second planning times on complex, non-convex trajectory-planning problems, such as the tractor-trailer parking task, highlight the potential of Safe MPD$^{\blacklozenge}$ to become a powerful tool for real-world autonomous systems. Our future work will focus on the deployment and validation of this framework on physical hardware.

REFERENCES

[1] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with Diffusion for Flexible Behavior Synthesis," in *International Conference on Machine Learning (ICML)*, 2022, pp. 9902–9915.

[2] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters, "Motion Planning Diffusion: Learning and Planning of Robot Motions with Diffusion Models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 1916–1923.

[3] C. Pan, Z. Yi, G. Shi, and G. Qu, "Model-Based Diffusion for Trajectory Optimization," in *Neural Information Processing Systems (NeurIPS)*, 2024, pp. 57 914–57 943.

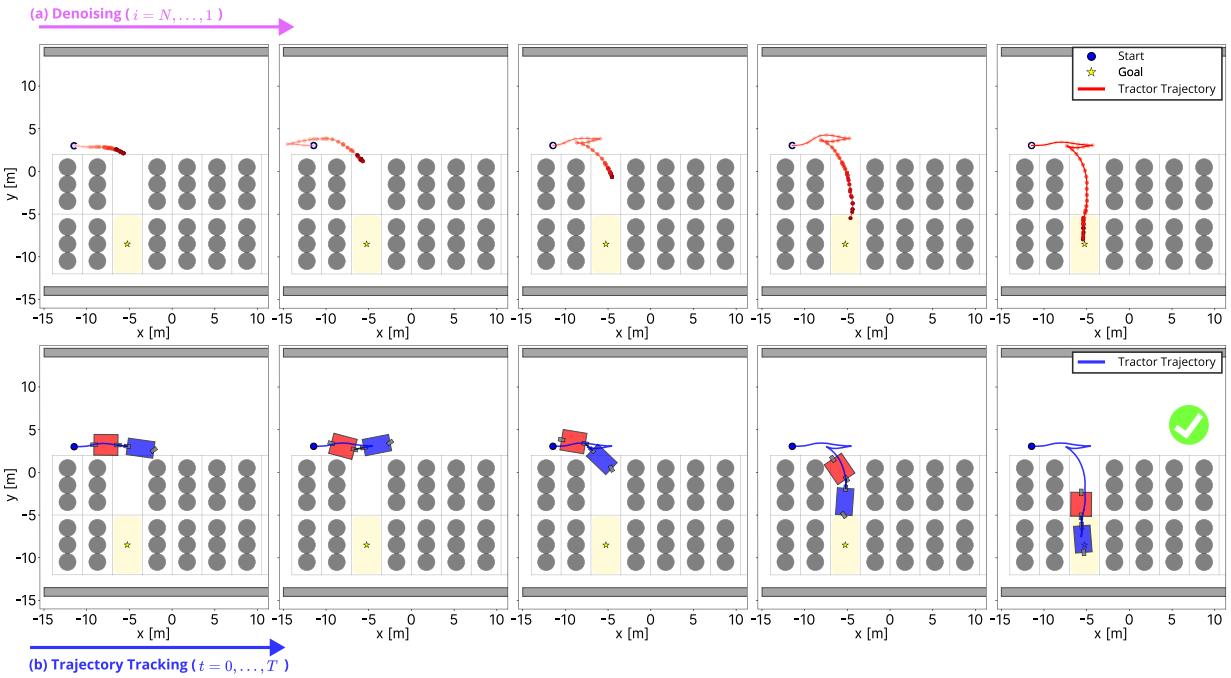

Fig. 3: Visualization of the diffusion process and final trajectory execution. (a) Snapshots of the diffusion trajectory for a kinematic tractor-trailer at different denoising steps, showing the refinement from a random, high-cost path (e.g., $i = 100$) to an optimized solution at the final step ($i = 0$). (b) Final trajectory with the tractor–trailer footprint rendered along the path; despite tight clearances, neither hitch-angle nor collision constraints are violated.

[4] T. Yun, S. Yun, J. Lee, and J. Park, "Guided Trajectory Generation with Diffusion Models for Offline Model-based Optimization," in *Neural Information Processing Systems (NeurIPS)*, 2024, pp. 83 847–83 876.

[5] Y. Qing, S. Chen, Y. Chi, S. Liu, S. Lin, K. Yao, and C. Zou, "BiTraJDiff: Bidirectional Trajectory Generation with Diffusion Models for Offline Reinforcement Learning," in *arXiv preprint arXiv.2506.05762*, 2025.

[6] Z. Zhong, D. Rempe, D. Xu, Y. Chen, S. Veer, T. Che, B. Ray, and M. Pavone, "Guided Conditional Diffusion for Controllable Traffic Simulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3560–3566.

[7] K. M. Lee, S. Ye, Q. Xiao, Z. Wu, Z. Zaidi, D. B. D'Ambrosio, P. R. Sanketi, and M. Gombolay, "Learning Diverse Robot Striking Motions with Diffusion Models and Kinematically Constrained Gradient Guidance," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 12 017–12 024.

[8] X. Huang, T. Truong, Y. Zhang, F. Yu, J. P. Sleiman, J. Hodgins, K. Sreenath, and F. Farshidian, "Diffuse-CLoC: Guided Diffusion for Physics-based Character Look-ahead Control," *ACM Transactions on Graphics*, vol. 44, no. 4, pp. 1–12, 2025.

[9] J. K. Christopher, S. Baek, and F. Fioretto, "Constrained Synthesis with Projected Diffusion Models," in *Neural Information Processing Systems (NeurIPS)*, 2024, pp. 89 307–89 333.

[10] J. K. Christopher, M. Cardei, J. Liang, and F. Fioretto, "Neuro-Symbolic Generative Diffusion Models for Physically Grounded, Robust, and Safe Generation," in *International Conference on Neuro-symbolic Systems (NeuS)*, 2025, pp. 188–213.

[11] R. Römer, A. v. Rohr, and A. P. Schoellig, "Diffusion Predictive Control with Constraints," in *Learning for Dynamics and Control (L4DC)*, 2025, pp. 791–803.

[12] H. Ma, S. Bodmer, A. Carron, M. Zeilinger, and M. Muehlebach, "Constraint-Aware Diffusion Guidance for Robotics: Real-Time Obstacle Avoidance for Autonomous Racing," in *arxiv preprint arXiv:2505.13131*, 2025.

[13] W. Xiao, T.-H. Wang, C. Gan, and D. Rus, "SafeDiffuser: Safe Planning with Diffusion Probabilistic Models," in *International Conference on Learning Representations (ICLR)*, 2025.

[14] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Neural Information Processing Systems (NeurIPS)*, 2020, pp. 6840–6851.

[15] R. Rubinstein, "The Cross-Entropy Method for Combinatorial and Continuous Optimization," *Methodology And Computing In Applied Probability*, vol. 1, no. 2, pp. 127–190, 1999.

[16] G. Williams, A. Aldrich, and E. A. Theodorou, "Model Predictive Path Integral Control: From Theory to Parallel Computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[17] J. Yin, C. Dawson, C. Fan, and P. Tsiotras, "Shield Model Predictive Path Integral: A Computationally Efficient Robust MPC Method Using Control Barrier Functions," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7106–7113, 2023.

[18] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition*, ser. Advances in Design and Control. Society for Industrial and Applied Mathematics, 2010.

[19] O. Bastani, "Safe Reinforcement Learning with Nonlinear Dynamics via Model Predictive Shielding," in *American Control Conference (ACC)*, 2021, pp. 3488–3494.

[20] D. R. Agrawal, R. Chen, and D. Panagou, "gatekeeper: Online Safety Verification and Control for Nonlinear Systems in Dynamic Environments," *IEEE Transactions on Robotics*, vol. 40, pp. 4358–4375, 2024.

[21] 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*, vol. 43, no. 5, pp. 137–177, 2023.

[22] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[23] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2019, pp. 2623–2631.

[24] K. Majd, H. Parwana, B. Hoxha, S. Hong, H. Okamoto, and G. Fainekos, "GPU-Accelerated Barrier-Rate Guided MPPI Control for Tractor-Trailer Systems," in *IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2025.

[25] H. Parwana, T. Kim, K. Long, B. Hoxha, H. Okamoto, G. Fainekos, and D. Panagou, "BR-MPPI: Barrier Rate guided MPPI for Enforcing Multiple Inequality Constraints with Learned Signed Distance Field," in *arXiv preprint arXiv:2506.07325*, 2025.