SAFEFLOWMATCHER: SAFE AND FAST PLANNING USING FLOW MATCHING WITH CONTROL BARRIER FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative planners based on Flow Matching (FM) produce high-quality paths in a single or a few ODE steps, but their sampling dynamics offer no formal safety guarantees and can yield incomplete paths near constraints. We present *SafeFlowMatcher*, a planning framework that couples FM with control barrier functions (CBFs) to achieve *both* real-time efficiency and certified safety. SafeFlowMatcher uses a two-phase *prediction–correction* (PC) integrator: (i) a prediction phase integrates the learned FM once (or a few steps) to obtain a candidate path without intervention; (ii) a correction phase refines this path with a vanishing time-scaled vector field and a CBF-based quadratic program that minimally perturbs the vector field. We prove a barrier certificate for the resulting flow system, establishing forward invariance of a robust safe set and finite-time convergence to the safe set. In addition, by enforcing safety only on the executed path—rather than all intermediate latent paths—SafeFlowMatcher avoids distributional drift and mitigates local trap problems. Moreover, SafeFlowMatcher attains faster, smoother, and safer paths than diffusion- and FM-based baselines on maze navigation and locomotion. Extensive ablations corroborate the contributions of the PC integrator and the barrier certificate.

1 INTRODUCTION

Robotic path planning must simultaneously achieve real-time responsiveness and strong safety guarantees. Recently, generative models such as diffusion (Ho et al., 2020; Dhariwal & Nichol, 2021b; Song et al., 2021b) and flow matching (FM) (Lipman et al., 2023) have gained attention for path planning, thanks to their expressive modeling of multi-modal action distributions (Carvalho et al., 2023; Braun et al., 2024) and low-latency inference (Qureshi et al., 2019; Liu et al., 2024) compared to classical sampling- and optimization-based planners. However, the sampling dynamics of these models are governed by implicitly learned rules and can produce paths that violate physical safety constraints, leading to task interruptions or collisions. Therefore, integrating *certified safety* into generative planning is essential for deployment in real-world robotic systems.

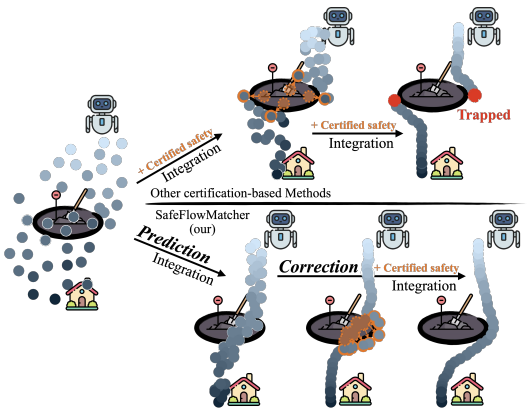

Figure 1: Directly constraining intermediate samples during generation (top) can cause paths to be distorted or trapped, whereas SafeFlowMatcher (bottom) decouples generation and certification, producing a complete and certified-safe path.

Several approaches have attempted to enforce safety in generative planning. Safety-guidance methods regulate the sampling process through learned safety scores (often called guidance, e.g., classifier(-free) guidance(Dhariwal & Nichol, 2021a; Ho & Salimans, 2021), or value/reward guidance (Yang et al., 2024; Chen et al., 2024)), but their reliance on data-driven proxy prevents them from providing strong safety guarantees. More ex-

plicit, certification-based methods incorporate functions such as Control Barrier Functions (CBFs) directly into the generative process (Wei et al., 2025). Unlike guidance-based approaches, these methods can guarantee safety at deployment without requiring additional training. However, a key challenge in such certification-based methods is a *semantic misalignment*: certification concerns the executed physical path (its waypoints over the horizon), whereas interventions are often applied to intermediate latent states that are never executed. Constraining such latents is unnecessary for certification. As a result, repeated interventions distort the learned flow and often yield incomplete (locally trapped) paths. Finally, although diffusion samplers can be accelerated (Lu et al., 2022; Zhang & Chen, 2022; Liu et al., 2022), their SDE-based denoising requires many steps, making real-time planning expensive. In contrast, FM casts sampling as deterministic ODE integration, generating accurate paths in a *single* or a *few* steps.

To address these limitations, we propose *SafeFlowMatcher*, a planning framework that combines flow matching with CBFs, particularly for finite-time convergence CBFs, to achieve certified safety before the completion of generation, while maintaining the efficiency of FM. Our key idea is a *Prediction–Correction* (PC) integrator that decouples distributional drift from safety certification. In the prediction phase, we propagate the flow once (or a few steps) to obtain a candidate path without any safety intervention. In the correction phase, we refine this path by (i) compensating for integration error through a modified vector field, and (ii) enforcing safety through CBFs. Rather than constraining all intermediate samples from pure noise to the target during prediction, SafeFlowMatcher enforces safety only in the correction phase. This preserves the native FM dynamics and prevents distributional drift when generating the target path. Also, it avoids local traps caused by repeatedly pushing intermediate waypoints onto the barrier boundary and stalling near safety constraints.

Our main contributions are as follows:

- We introduce SafeFlowMatcher, a novel planning framework that integrates finite-time convergence CBF-based certification with flow matching to enforce hard safety constraints, while preserving the efficiency of flow matching.
- We propose prediction–correction integrator that decouples path generation from certification: FM first generates paths without intervention, and then CBF-based corrections enforce finite-time convergence to the safe set while compensating for integration errors.
- We validate SafeFlowMatcher in maze navigation and locomotion with extensive ablation studies, showing consistent improvements over both FM- and diffusion-based planners in efficiency, safety, and path quality.

2 Related Work & Preliminaries

2.1 FlowMatcher: Flow Matching for Planning

FM has recently been proposed as a powerful alternative to diffusion, originally in the image generation domain (Lipman et al., 2023; Song et al., 2021b), and has shown promise for efficient path planning and robotic control (Ye & Gombolay, 2024; Zhang & Gienger, 2024; Chisari et al., 2024; Xing et al., 2025). Unlike diffusion, FM directly learns a time-varying vector field that maps noise to the target distribution via forward integration, making the sampling process efficient and flexible.

We adapt standard flow matching (FM) (Lipman et al., 2023) to planning context. Let $H \in \mathbb{N}$ be the planning horizon and $\mathcal{H} \triangleq \{0, \dots, H\}$. A path is a stacked vector $\boldsymbol{\tau} = (\boldsymbol{\tau}^0, \boldsymbol{\tau}^1, \dots, \boldsymbol{\tau}^H) \in \mathcal{D}^{H+1} \subseteq \mathbb{R}^{d\times(H+1)}$, where each waypoint $\boldsymbol{\tau}^k \in \mathcal{D} \subseteq \mathbb{R}^d$ encodes the state at step k.

Let $v_t(\cdot;\theta) : \mathcal{D}^{H+1} \to \mathcal{D}^{H+1}$ be a time-dependent vector field. The flow $\psi : [0,1] \times \mathcal{D}^{H+1} \to \mathcal{D}^{H+1}$ is defined as the solution of the ODE

$$\frac{d}{dt}\,\psi_t(\boldsymbol{\tau}) = v_t(\psi_t(\boldsymbol{\tau});\theta), \qquad \psi_0(\boldsymbol{\tau}) = \boldsymbol{\tau}, \tag{1}$$

which transports a simple prior p_0 (e.g. $\mathcal{N}(0, I)$) to a target p_1. Following conditional flow matching (CFM), we train $v_t(\cdot;\theta)$ by regressing it to a conditional vector field that generates a fixed conditional probability path. We adopt the optimal transport (OT) path $p_t(\boldsymbol{\tau} \mid \boldsymbol{\tau}_1) = \mathcal{N}\big(\boldsymbol{\tau};\, \mu_t(\boldsymbol{\tau}_1),\, \sigma_t^2 I\big),\ \mu_t(\boldsymbol{\tau}_1) = t\,\boldsymbol{\tau}_1,\ \sigma_t = 1 - t$, whose generating *OT-conditional vector field* is

$$u_t(\boldsymbol{\tau} \mid \boldsymbol{\tau}_1) = \frac{\boldsymbol{\tau}_1 - \boldsymbol{\tau}}{1 - t}. \tag{2}$$

Sampling $t \sim \mathrm{Unif}[0,1]$, $\boldsymbol{\tau}_0 \sim p_0$, $\boldsymbol{\tau}_1 \sim q$ and defining $\boldsymbol{\tau}_t \triangleq \psi_t(\boldsymbol{\tau}_0) = (1-t)\boldsymbol{\tau}_0 + t\,\boldsymbol{\tau}_1$ (conditioned on $\boldsymbol{\tau}_1$), we have by (2) that $u_t(\boldsymbol{\tau}_t \mid \boldsymbol{\tau}_1) = \boldsymbol{\tau}_1 - \boldsymbol{\tau}_0$. Hence, we train $v_t(\cdot;\theta)$ with the CFM loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,\, q(\boldsymbol{\tau}_1),\, p_0(\boldsymbol{\tau}_0)} \big\| \, v_t(\psi(\boldsymbol{\tau}_0);\theta) - (\boldsymbol{\tau}_1 - \boldsymbol{\tau}_0) \, \big\|_2^2. \tag{3}$$

Further details are in Lipman et al. (2023).

For numerical integration, we discretize $0 = t_0 < \cdots < t_T = 1$ with $T \in \mathbb{N}$ (Collectively $\mathcal{T}(T) = \{t_0, \ldots, t_T\}$) and define step sizes $\Delta t_i = t_{i+1} - t_i$. We define T-step integrator $\Psi_{0\to 1}^{(T)} : \mathcal{D}^{H+1} \to \mathcal{D}^{H+1}$ (e.g., Euler integrator [1]) which integrates the flow matching dynamics from $\boldsymbol{\tau}_0$ to $\boldsymbol{\tau}_1$ as

$$\Psi_{0\to 1}^{(T)}(\boldsymbol{\tau}_0) = \boldsymbol{\tau}_0 + \sum_{i=0}^{T-1} \Delta t_i \, v_{t_i}(\boldsymbol{\tau}_{t_i};\theta). \tag{4}$$

2.2 Control Barrier Functions

Safety filters (Hsu et al., 2023; Wabersich et al., 2023) are a real-time intervention mechanism to ensure that an autonomous agent operates within some predefined safety sets, overriding its nominal behavior only when it is about to violate the sets. Various approaches exist for constructing safety filters, including reachability-based methods (Bansal et al., 2017), model predictive control (Hewing et al., 2020; Wabersich & Zeilinger, 2021), and learning-based safety critics (Alshiekh et al., 2018; Srinivasan et al., 2020). Among these, *control barrier functions (CBFs)* (Ames et al., 2019) are especially popular as they provide a systematic way to guarantee forward invariance of safe sets by solving a real-time optimization problem at each control step. They have wide application in autonomous driving (Ames et al., 2016), legged locomotion (Kim et al., 2023), multi-robot systems (Wang et al., 2017), and more. Here, we review only the CBF preliminaries that are necessary for the rest of this paper; the reader is referred to the above references for more details.

We consider systems of the following control-affine form

$$\dot{\mathbf{x}}_t = f(\mathbf{x}_t) + g(\mathbf{x}_t)\mathbf{u}_t, \tag{5}$$

where $\mathbf{x}_t \in \mathcal{D} \subset \mathbb{R}^d$, $\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^m$, and $f : \mathbb{R}^d \to \mathbb{R}^d$ and $g : \mathbb{R}^{d\times m} \to \mathbb{R}^d$ are locally Lipschitz continuous.

Define the *safe set* $\mathcal{C}$ as the superlevel set of a continuously-differentiable (C^1) function $b : \mathcal{D} \to \mathbb{R}$,

$$\mathcal{C} \triangleq \{\mathbf{x}_t \in \mathcal{D} \mid b(\mathbf{x}_t) \geq 0\}. \tag{6}$$

Definition 1 (Finite-Time Convergence CBF) *Given the system (5) and the safe set (6), C^1 function b is called a finite-time convergence CBF if there exist parameters $\rho \in [0,1)$ and $\epsilon > 0$ such that for all $\mathbf{x}_t \in \mathcal{D}$,*

$$\sup_{\mathbf{u}_t \in \mathcal{U}} [L_f b(\mathbf{x}_t) + L_g b(\mathbf{x}_t)\mathbf{u}_t + \epsilon \cdot \mathrm{sgn}(b(\mathbf{x}_t))|b(\mathbf{x}_t)|^\rho] \geq 0, \tag{7}$$

where $L_f b(\mathbf{x}_t) \triangleq \nabla b(\mathbf{x}_t)^\top f(\mathbf{x}_t)$ and $L_g b(\mathbf{x}_t) \triangleq \nabla b(\mathbf{x}_t)^\top g(\mathbf{x}_t)$ denote the Lie derivatives of b along f and g, respectively.

Lemma 1 (Forward Invariance of the Safe Set) *Define CBF b as in Definition 1, such that the initial state satisfies $b(\mathbf{x}_0) \geq 0$. Any Lipschitz continuous controller $\mathbf{u}_t$ that satisfies condition (7) ensures forward invariance of the safe set $\mathcal{C}$, i.e., $b(\mathbf{x}_t) \geq 0$ for all $t \geq 0$.*

To select a control input that guarantees forward invariance of $\mathcal{C}$ as well as become close as possible to some reference control input $\mathbf{u}_t^{\text{ref}}$, a common approach is to solve a quadratic program (CBF-QP) (Ames et al., 2019) at each time step:

$$\mathbf{u}_t^* = \underset{\mathbf{u}_t \in \mathcal{U}}{\arg\min} \ \|\mathbf{u}_t - \mathbf{u}_t^{\text{ref}}\|^2 \quad \text{subject to} \quad L_f b(\mathbf{x}_t) + L_g b(\mathbf{x}_t)\mathbf{u}_t + \epsilon \cdot \mathrm{sgn}(b(\mathbf{x}_t))|b(\mathbf{x}_t)|^\rho \geq 0. \tag{8}$$

This means the optimal solution $\mathbf{u}_t^*$ is the minimally modified control that guarantees the forward invariance of the safe set $\mathcal{C}$. Moreover, based on the finite-time stability theorem (Bhat & Bernstein, 2000), the finite-time convergence CBF can be used to ensure that states not only remain within the safe set but also reach it within finite-time (Li et al., 2018; Srinivasan et al., 2018).

[1] Alternatively, higher-order ODE solvers can be used.

3 SAFEFLOWMATCHER

Here, we present *SafeFlowMatcher*, a safe and fast planning framework that couples flow matching with certified safety. First, in Section 3.1, we introduce a two-phase *Prediction-Correction* (PC) integrator which decouples generation and certification. Next, in Section 3.2, we formalize safety for SafeFlowMatcher by employing control barrier functions (CBFs) and derive conditions that guarantee forward invariance and finite-time convergence to the safe set. The pseudocode of SafeFlowMatcher is in Algorithm 1.

We introduce a crucial problem in non-autoregressive planner, particularly for generative-based planner. As shown in Figure 2, non-autoregressive planners may fail to generate a complete path after planning when using CBFs. Although the resulting path remains safe (does not exceed safety constraints), it may be unable to reach the goal because certain waypoints become *locally trapped* near the barrier boundaries and cannot escape within the finite sampling or integration time. We will show that SafeFlowMatcher can effectively resolve this issue using a PC integrator.

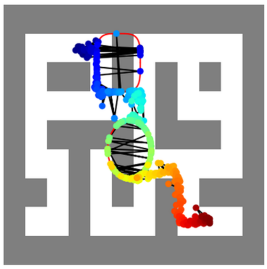

Figure 2: Example of a local trap in maze environment.

Definition 2 (Local Trap) *A local trap problem occurs during the planning process if there exists $k \in \mathcal{H}$ such that $\|\boldsymbol{\tau}_1^k - \boldsymbol{\tau}_1^{k-1}\| > \zeta$, where $\zeta > 0$ is a user-defined threshold depending on the planning environment.* [2]

Algorithm 1 SafeFlowMatcher

Input: learned velocity field $v_t(\cdot;\theta)$, prediction and correction time horizon T^p, T^c, planning horizon $\mathcal{H}$, CBF parameters (ϵ, ρ), robustness parameter δ, and scale constant α
Output: Safe path $\boldsymbol{\tau}_1^c$
Phase 1: Prediction
Sample initial noise $\boldsymbol{\tau}_0^p \sim \mathcal{N}(0, I)$
Compute predicted path $\boldsymbol{\tau}_1^p \leftarrow \Psi_{0\to1}^{(T^p)}(\boldsymbol{\tau}_0^p)$ by (9)
Phase 2: Correction
Initialize corrected path $\boldsymbol{\tau}_0^c \leftarrow \boldsymbol{\tau}_1^p$
for each correction step $t \in \mathcal{T}(T^c)$ **do**
  for each waypoint $k \in \mathcal{H}$ **do**
    Solve QP (16) to obtain $(\mathbf{u}_t^{k*}, r_t^{k*})$
  end for
  Update velocity time-scaled flow dynamics (13) with $\mathbf{u}_t^* = \{\mathbf{u}_t^{0,*}, \ldots, \mathbf{u}_t^{H,*}\}$
end for
Return safe final path $\boldsymbol{\tau}_1^c$

3.1 PREDICTION-CORRECTION INTEGRATOR

SafeFlowMatcher divides the integration process into two phases: a prediction phase that generates an approximate path without considering safety, and a correction phase that refines the path by reducing integration error and adding safety constraints. Let $\boldsymbol{\tau}_t^\ell \in \mathcal{D}^{H+1} \subseteq \mathbb{R}^{d\times(H+1)}$ for $\ell \in \{p, c\}$ denote the paths in the prediction and correction phases, with waypoints $\boldsymbol{\tau}_t^{\ell,k} \in \mathcal{D} \subseteq \mathbb{R}^d$ for $k \in \mathcal{H}$.

Prediction phase aims to quickly approximate the target path starting from pure noise $\boldsymbol{\tau}_0^p \sim \mathcal{N}(0, I)$, without considering safety constraints. Starting from the noise, we run Euler integration to obtain the solution of the flow matching dynamics (1):

$$\boldsymbol{\tau}_1^p = \Psi_{0\to1}^{(T)}(\boldsymbol{\tau}_0^p) = \boldsymbol{\tau}_1^\star + \varepsilon, \tag{9}$$

where $\boldsymbol{\tau}_1^\star$ is the exact solution of the flow matching dynamics and ε is the Euler integration (prediction) error. To balance computational efficiency and reliability, we select small T (typically $T = 1$) that places $\boldsymbol{\tau}_0^p$ sufficiently close to $\boldsymbol{\tau}_1^\star$, making it a suitable initialization for the correction phase.

[2] The definition is slightly different from that of SafeDiffuser (Wei et al., 2025) to capture a broader class of failure cases. See Appendix C for the details.

Correction phase starts from the path in the prediction phase $\boldsymbol{\tau}_0^c = \boldsymbol{\tau}_1^p$, unlike $\boldsymbol{\tau}_0^p$ in the prediction phase. In this phase, the path is refined by (i) decreasing the discretization error ε and (ii) enforcing safety constraints.

We define a new flow dynamics, called *vanishing time-scaled flow dynamics* (VTFD), with scale constant $\alpha \geq 1$:

$$\frac{d\boldsymbol{\tau}_t^c}{dt} = \alpha\,(1-t)\,v_t(\boldsymbol{\tau}_t^c;\,\theta) \triangleq \tilde{v}_t(\boldsymbol{\tau}_t^c;\theta), \tag{10}$$

where $\tilde{v}_t(\boldsymbol{\tau}_t^c;\theta)$ denotes the vanishing time-scaled vector field. The $(1-t)$ factor gradually suppresses the dynamics as $t \to 1$, ensuring that the path stabilizes close to the target rather than drifting.

Lemma 2 *Assume the prediction error $\varepsilon \sim p_\varepsilon$ has a symmetric, zero-mean distribution (e.g., Gaussian) and that, in a neighborhood of $\varepsilon = 0$, the negative log-density $-\log p_\varepsilon$ is C^2 with a positive-definite Hessian $A \succ 0$ (i.e., locally strongly convex). In addition, assume the target log-density $\log p_1$ is C^2. Suppose the correction phase is initialized near the target $\boldsymbol{\tau}_1^\star$:*

$$\boldsymbol{\tau}_t^c = \boldsymbol{\tau}_1^\star + (1-t)\,\varepsilon, \qquad \varepsilon = O(1). \tag{11}$$

Then, $\mathbb{E}[\boldsymbol{\tau}_1 \mid \boldsymbol{\tau}_t^c] = \boldsymbol{\tau}_1^\star + O(1-t)$.

(11) is a natural assumption under optimal transport, since OT path approaches $\boldsymbol{\tau}_1$ as $t \to 1$. Lemma 2 ensures expectation of posterior contract towards target. The next lemma implies that VTFD can reduce the prediction error.

Lemma 3 *Under the assumptions of Lemma 2, let $\mathbf{e}_t \triangleq \boldsymbol{\tau}_t^c - \boldsymbol{\tau}_1^\star$. If the flow dynamics follow the vanishing time-scaled flow dynamics (10), then as $t \to 1$,*

$$\mathbf{e}_t = O((1-t)^2) + (\,\varepsilon + O(1)\,)e^{-\alpha t}. \tag{12}$$

Lemma 3 implies that $\boldsymbol{\tau}_1^c$ has only a tiny error. See proofs of Lemma 2 and Lemma 3 in Appendix A.

3.2 Control Barrier Certificate for SafeFlowMatcher

To ensure that the generated waypoint $\boldsymbol{\tau}_t^{c,k}$ escapes from unsafe set $\mathcal{D}\backslash\mathcal{C}$ and remains within safe set $\mathcal{C}$, we introduce an additional perturbation to minimally intervene the flow matching dynamics (1):

$$\frac{d\boldsymbol{\tau}_t^c}{dt} = \tilde{v}_t(\boldsymbol{\tau}_t^c;\theta) + \Delta\mathbf{u}_t \triangleq \mathbf{u}_t, \tag{13}$$

where $\mathbf{u}_t = \{\mathbf{u}_t^0, \mathbf{u}_t^1, ..., \mathbf{u}_t^H\} \in \mathcal{D}^{H+1}(\mathbf{u}_t^k \in \mathcal{D})$, and $\Delta\mathbf{u}_t \in \mathbb{R}^{d\times(H+1)}$ is a perturbation term that enforces safety constraints. For notational simplicity, we denote the right-hand side by $\mathbf{u}_t \in \mathbb{R}^{d\times(H+1)}$. The original equation can be equivalently substituted in the following derivations without affecting the results. We now formalize the concept of safety in flow matching using finite-time flow invariance.

Definition 3 (Finite-Time Flow Invariance) *Let $b : \mathcal{D} \to \mathbb{R}$ be a C^1 function. The system (13) is finite-time flow invariant if there exists $t_f \in [0,1]$ such that $b(\boldsymbol{\tau}_t^{c,k}) \geq 0$ for all $k \in \mathcal{H}$, $\forall t \geq t_f$.*

Theorem 1 (Forward Invariance for SafeFlowMatcher) *Let $b : \mathcal{D} \to \mathbb{R}$ be a C^1 function, and define the robust safety set $\mathcal{C}_\delta \triangleq \{\boldsymbol{\tau}^{c,k} \in \mathcal{D} \mid b(\boldsymbol{\tau}^{c,k}) \geq \delta\}$ for some $\delta > 0$. Suppose the system (13) is controlled by $\mathbf{u}_t$ satisfying the following barrier certificate for $0 < \rho < 1$, $\epsilon > 0$:*

$$\nabla b(\boldsymbol{\tau}_t^{c,k})^\top \mathbf{u}_t^k + \epsilon \cdot \mathrm{sgn}(b(\boldsymbol{\tau}_t^{c,k}) - \delta)|b(\boldsymbol{\tau}_t^{c,k}) - \delta|^\rho + w_t^k r_t^k \geq 0, \forall k \in \mathcal{H}, \forall t \in [0,1]. \tag{14}$$

Here, $w_t^k : [0,1] \to \mathbb{R}_{\geq 0}$ is a monotonically decreasing function with $w_t^k = 0$ for all $t \in [t_w, 1]$ ($t_w \in [0,1)$), and $r_t^k \geq 0$ is a slack variable. Then the flow matching (13) achieves finite-time flow invariance on $\mathcal{C}_\delta$.

The weights w_t^k serve as functions that relax the CBF constraint in the early refining phase, providing numerical stability by preventing infeasibility and reducing abrupt changes in the QP solution. Since w_t^k vanishes for $t \geq t_w$, the relaxation term has no effect afterwards, ensuring that the final path satisfies certified safety.

Proposition 1 (Finite Convergence Time for SafeFlowMatcher) *Suppose Theorem 1 holds. Then for any initial path $\boldsymbol{\tau}_{t_w}^{c,k} \in \mathcal{D} \setminus \mathcal{C}_\delta$, the state path $\boldsymbol{\tau}_t^{c,k}$ converges to the safe set $\mathcal{C}_\delta$ within finite time*

$$T \leq t_w + \frac{(\delta - b(\boldsymbol{\tau}_{t_w}^{c,k}))^{1-\rho}}{\epsilon(1-\rho)}, \tag{15}$$

and remains in the set thereafter.

Proposition 1 allows us to select parameters ϵ and ρ to guarantee flow invariance on the robust safe set $\mathcal{C}_\delta$ before the time (15). The proofs of Theorem 1 and Proposition 1 are in Appendix B

In order to enforce the invariance of the safety set $\mathcal{C}_\delta$ with minimum intervention during planning, we solve a quadratic program (QP) analogous to (8) at each sampling time t and planning step k:

$$\mathbf{u}_t^{k*}, r_t^{k*} = \underset{\mathbf{u}_t^k, r_t^k}{\arg\min} \|\mathbf{u}_t^k - \tilde{v}_t^k(\boldsymbol{\tau}_t^c; \theta)\|^2 + {r_t^k}^2 \quad \text{subject to} \quad (14), \tag{16}$$

where $\tilde{v}_t^k(\boldsymbol{\tau}_t^c; \theta)$ denotes the k-th column of $\tilde{v}_t(\boldsymbol{\tau}_t^c; \theta)$. Since the QP (16) is equivalent to a Euclidean projection problem with linear inequalities, closed-form solutions are available when it has at most two inequalities (Luenberger, 1997; Boyd & Vandenberghe, 2004).

Remark 1 *The Prediction–Correction integrator brings $\boldsymbol{\tau}_0^c$ closer to the barrier boundary after the prediction phase, and further reduces $b(\boldsymbol{\tau}_{t_w}^c)$ during correction. By Proposition 1, this tighter initialization shortens the required convergence time, allowing us a wider range of choices for (ρ, ϵ), and more stable control inputs.*

4 Experiments

We evaluate SafeFlowMatcher through experiments designed to answer three key questions:

1. Does SafeFlowMatcher outperform state-of-the-art generative model based safe planning baselines in terms of safety, planning performance, and efficiency?
2. Does SafeFlowMatcher really require a two-phase (prediction and correction) approach?
3. How well can SafeFlowMatcher generalize to more complex tasks (e.g., robot locomotion)?

We conduct experiments on a variety of planning domains: (i) Maze navigation (`maze-large-v1`), (ii) OpenAI Gym locomotion (`Walker2D-Medium-Expert-v2`, `Hopper-Medium-Expert-v2`) (Brockman et al., 2016; Todorov et al., 2012).

To fairly evaluate our proposed method, we extend *SafeDiffuser* (Wei et al., 2025) beyond its original DDPM sampler. We introduce three additional safety-aware variants. For the first and second variants, we adapt DDIM (Song et al., 2021a) into two versions, *SafeDDIM(η=0.0 & 1.0)*, which share the same training settings as SafeDiffuser; here, η controls the level of sampling randomness. The last variant we develop is *SafeFM*, a flow-matching counterpart to SafeDiffuser which uses the same model size and training setup as SafeFlowMatcher, but enforces safety directly during sampling and without the prediction–correction step. For all flow-matching based methods, including *SafeFM* and *SafeFlowMatcher*, we apply a simple covariance-aware guidance $g^{\text{cov-A}}$ with scale 1.0, following prior work (Feng et al., 2025). When safety constraints are disabled, we drop the "Safe" prefix. Additional details on experimental settings are provided in Appendix D.1.

For safety, we report *Barrier Safety* (BS) per constraint, the minimum value of the barrier function b (which should remain non-negative), and *Trap Rate*, the rate of local trap occurrences. For planning quality, we measure the overall *Score*, path *Curvature* (κ) averaged over the planning horizon, and path *Acceleration* (a) averaged over the horizon. For efficiency, we report *Time*, the computation time per path in milliseconds. Formal definitions of the metrics are provided in Appendix D.2.

4.1 Main Results on Maze2D Navigation

We first present the main performance comparison in the Maze2D setting, as shown in Figure 4, where there are two safety constraints (red circles). As shown in Table 1 and Figure 3, SafeFlowMatcher demonstrates a superior trade-off across all metrics in the Maze2D environment. It achieves

Method	BS1(↑) (≥ 0)	BS2(↑) (≥ 0)	Score(↑)	TIME (ms)	TRAP RATE	$\kappa(\downarrow)$	$a(\downarrow)$
Diffuser (Janner et al., 2022)	-0.825	-0.784	1.572±0.288	3.70	0%	77.04±4.30	86.68±3.81
Truncation (Brockman et al., 2016)	-0.999	-0.999	0.978±0.128	19.51	100%	1118.21±1093.96	9.043e5±8.988e6
CG (Dhariwal & Nichol, 2021b)	-0.996	-0.999	0.505±0.092	19.13	100%	949.63±1103.62	959.71±1846.58
CG-ϵ (Dhariwal & Nichol, 2021b)	-0.998	-0.999	0.499±0.104	19.87	100%	1027.28±1124.70	1.202e9±1.1961e10
ROS-SafeDiffuser (Wei et al., 2025)	0.010	0.010	1.435±0.502	4.67	100%	75.15±6.67	422.87±86.70
RES-SafeDiffuser (Wei et al., 2025)	0.010	0.010	1.442±0.451	4.72	72%	80.30±13.06	398.17±1060.86
TVS-SafeDiffuser (Wei et al., 2025)	-0.003	-0.003	1.506±0.405	4.78	69%	78.72±7.80	124.51±34.22
ROS-SafeDDIM($\eta = 0.0$)	0.010	0.010	1.132±0.556	4.79	100%	31.22±4.87	2073.84±1694.06
RES-SafeDDIM($\eta = 0.0$)	0.010	0.010	1.405±0.494	4.83	96%	43.23±3.41	1153.81±2040.98
TVS-SafeDDIM($\eta = 0.0$)	-0.026	-0.026	1.522±0.295	4.79	90%	42.56±3.39	575.73±371.83
ROS-SafeDDIM($\eta = 1.0$)	0.010	0.010	1.575±0.158	4.89	100%	56.30±2.93	668.17±69.19
RES-SafeDDIM($\eta = 1.0$)	0.010	0.010	1.532±0.331	4.82	86%	61.73±4.80	1584.00±8085.06
TVS-SafeDDIM($\eta = 1.0$)	-0.026	-0.026	1.549±0.304	4.74	65%	60.29±3.41	27.23±43.20
ROS-SafeFM	0.010	0.010	1.138±0.556	4.68	100%	23.57±8.34	1.317e4±9.931e4
RES-SafeFM	0.010	0.010	1.401±0.429	4.74	12%	61.17±19.52	6724.64±5.304e4
TVS-SafeFM	-0.002	-0.002	1.350±0.417	4.73	41%	60.29±3.41	768.71±2212.17
SafeFlowMatcher(ours)	0.010	0.010	1.632±0.003	4.71	0%	69.19±1.02	91.90±0.77

Table 1: **Performance comparison of different methods.** we evaluated all methods over 100 independent trials under identical settings. SafeFlowMatcher exhibits consistently low variance across metrics and maintains a zero trap rate. All baselines are reproduced by us.

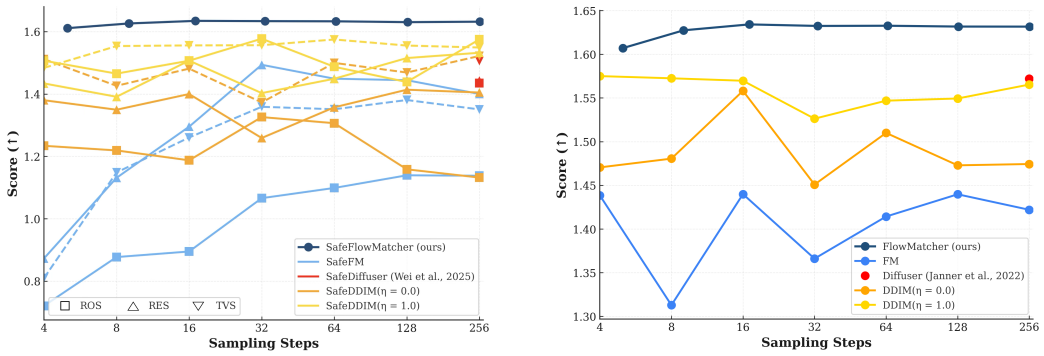

Figure 3: **Score versus number of sampling(integration) steps.** Left (safety on): SafeFlowMatcher attains the highest score across sampling steps. Right (safety off): FlowMatcher (FM + PC integrator) remains more sampler-efficient than others. Overall, SafeFlowMatcher outperforms in terms of score across all sampling steps.

the highest score while preserving safety, with comparably low computation time. Our method also attains top scores across all sampling steps.

Figure 4 provides a visual comparison, illustrating that SafeFlowMatcher generates smooth, efficient paths that effectively avoid obstacles, whereas baselines may produce unsafe, suboptimal, or computationally expensive paths.

4.2 Ablation Studies: Why Two Phases Are Important

Effectiveness of the Two-Phase Prediction-Correction Architecture. To emphasize the necessity of both the Prediction and the Correction phases in SafeFlowMatcher, we consider ablation studies with each phase removed. The "Prediction-Only" variant suffers from low safety, while the "Naive Safe FM" approach, which applies safety constraints from the beginning (like a "Correction-Only" variant), struggles to find successful paths, resulting in a high trap rate. Our full model successfully combines the strengths of both, achieving high task performance and safety.

Analysis on Number of Function Evaluations (NFE) Allocation. We analyze the trade-off between the number of function evaluations (NFE) allocated to the prediction phase (T^p) and the correction phase (T^c). Table 2 reports the score after both phases for varying T^p, and Figure 5 visualizes how T^p shapes the prediction paths. Increasing T^p yields slightly smoother predictions, while incurring extra computation. A single-step prediction (T^p=1) already suffices to initialize the

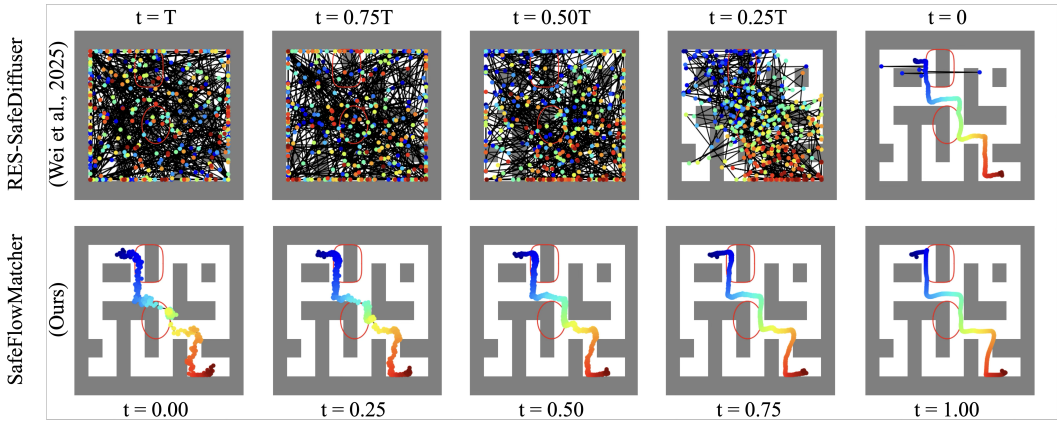

Figure 4: **Comparisons of the path generation process in Maze2D.** Red circles indicate the safety constraints the path should satisfy. (Top) RES-SafeDiffuser initializes samples all over the maze and converges to a path that has local traps. (Bottom) SafeFlowMatcher (ours) initializes from near target path after prediction phase, and converges to a higher-quality path with no local traps.[3]

corrector effectively and matches the performance of substantially larger T^p (e.g., 16). We therefore adopt T^p=1 as the default, balancing performance and compute.

Table 2: **Score versus T^p**. Score after both prediction and correction phases over T^p.

Prediction NFE (T^p)	**1**	**2**	**4**	**8**	**16**
score (↑)	1.632±0.008	1.520±0.340	1.468±0.434	1.362±0.480	1.632±0.003

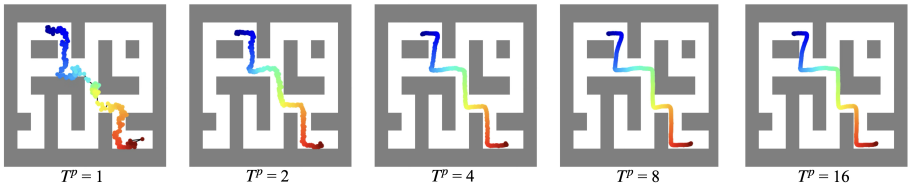

Figure 5: **Predicted path quality versus T^p**. The path at the end of the prediction phase for the Maze2D environment over T^p.

Importance of a Vanishing Scaling for Stable Correction. We empirically validate the role of the vanishing time–scaled flow dynamics (10) used in the correction phase. We sweep the scaling constant α and report scores in Table 3. Moderate scaling constant ($\alpha = 2$) yields the best trade-off: larger α accelerates early convergence but can over-amplify the vector field and reduce stability, while smaller α cannot effectively reduce error.

Table 3: **Score versus scaling constant in correction phase.**

Scaling constant α	**1.0**	**1.5**	**2.0**	**2.5**	**3.0**
Score (↑)	1.623±0.005	1.629±0.004	1.632±0.008	1.618±0.033	1.572±0.058

In Figures 6 and 7, we compare correction with and without vanishing time scaling. Figure 6 shows that, as sampling steps increases, vanishing time-scaling helps to prevent score reduction. In Figure 7, the path without a vanishing scaling exhibits a sharp drift near $t = 1$ and a larger gap between $\boldsymbol{\tau}_0^c$ and $\boldsymbol{\tau}_1^c$. These observations indicate that the vanishing time–scaled flow dynamics bounds the effective velocity near t=1, ensuring stable integration even as the sampling budget increases.

[3]Diffusion-based samplers evolve backward on an interval $[0, T]$, whereas flow matching evolves forward on $[0, 1]$; a natural correspondence can be established by normalizing $T = 1$ and reversing time.

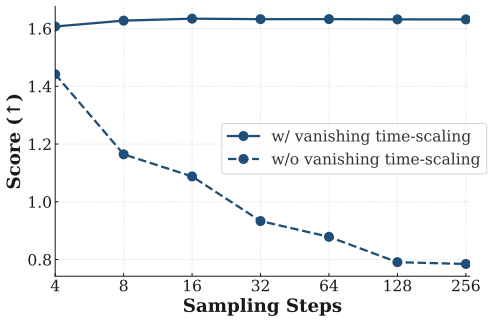

Figure 6: **Score with and without a vanishing time-scale.** As the sampling steps increases, we see that the score decreases in the absence of vanishing time-scale.

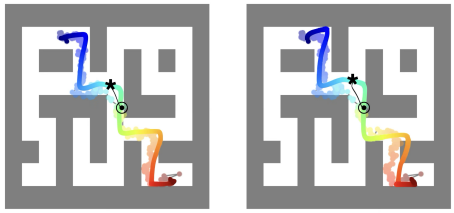

Figure 7: **Path quality with (a) and without (b) a vanishing time-scale.** The transparent path represents $\boldsymbol{\tau}_0^c$, the solid path represents $\boldsymbol{\tau}_1^c$. The black solid line represents the trajectory $\boldsymbol{\tau}_t^c$ from to $*$ over the interval $t \in [0, 1]$.

4.3 Generalization to High-Dimensional Robotic Tasks

We assess generalization to high-dimensional control on Walker2D and Hopper. Across both tasks, SafeFlowMatcher attains the highest score while maintaining BS ≥ 0, indicating that the prediction–correction design scales beyond static maze navigation. Note that the BS metric here is reported in a different way than in Table 1; here, BS is a binary indicator (yes or no) of whether safety is guaranteed (≥ 0) or not (< 0). We only compared methods that directly incorporate safety guarantees, hence BS≥ 0 is true for all of them, but this is not true of the other baselines we've considered so far (see Table 1). Due to space limitations, we defer all details on the experiment setup and additional algorithmic modifications to Appendix D.1.

Table 4: **Performance on high-dimensional robotic tasks.** SafeFlowMatcher maintains its advantages in safety and score even for more complex settings.

Environment	Method	Score (↑)	BS (≥ 0)
Walker2D	SafeDiffuser (Wei et al., 2025)	0.321 ± 0.119	Yes
	SafeFM	0.264 ± 0.127	Yes
	Ours	0.331 ± 0.021	**Yes**
Hopper	SafeDiffuser (Wei et al., 2025)	0.464 ± 0.028	Yes
	SafeFM	0.675 ± 0.312	Yes
	Ours	0.917 ± 0.026	**Yes**

5 Conclusion

We introduced *SafeFlowMatcher*, a planning framework that couples flow matching (FM) with CBF-certified safety by employing a two-phase *prediction–correction* integrator. On the path generation side, we proposed the vanishing time-scaled flow dynamics, which contracts the prediction error toward the target path. On the safety side, we established a finite convergence time barrier certificate for the flow system to ensure forward invariance of a safe set. The approach generates a candidate path with the learned FM dynamics and then refines only the *executed* path under safety constraints. This decoupling preserves the native generative dynamics, avoids distributional drift from repeated interventions on latent states, and mitigates local trap failures near constraint boundaries. Empirically, SafeFlowMatcher attains faster, smoother, and safer paths than various diffusion- and FM-based baselines across maze navigation and robot locomotion tasks. Some directions of future work include a more adaptive method of fine-tuning our hyperparameters, and the development of SafeFlowMatcher without guidance or conditioning.

REFERENCES

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8): 3861–3876, 2016.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pp. 3420–3431. Ieee, 2019.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2242–2253. IEEE, 2017.

Sanjay P Bhat and Dennis S Bernstein. Finite-time stability of continuous autonomous systems. *SIAM Journal on Control and optimization*, 38(3):751–766, 2000.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Max Braun, Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Riemannian flow matching policy for robot motion learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5144–5151. IEEE, 2024.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

Joao Carvalho, An T Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1916–1923. IEEE, 2023.

Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion. *CoRR*, abs/2401.02644, 2024.

Eugenio Chisari, Nick Heppert, Max Argus, Tim Welschehold, Thomas Brox, and Abhinav Valada. Learning robotic manipulation policies from point clouds with conditional flow matching. In *8th Annual Conference on Robot Learning*, 2024.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 8780–8794. Curran Associates, Inc., 2021a.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021b.

Ruiqi Feng, Chenglei Yu, Wenhao Deng, Peiyan Hu, and Tailin Wu. On the guidance of flow matching. In *Forty-second International Conference on Machine Learning*, 2025.

Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):269–296, 2020.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

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

Kai-Chieh Hsu, Haimin Hu, and Jaime 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.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022.

Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

Jeeseop Kim, Jaemin Lee, and Aaron D Ames. Safety-critical coordination for cooperative legged locomotion via control barrier functions. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2368–2375. IEEE, 2023.

Anqi Li, Li Wang, Pietro Pierpaoli, and Magnus Egerstedt. Formally correct composition of coordinated behaviors using control barrier certificates. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3723–3729. IEEE, 2018.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Jianwei Liu, Maria Stamatopoulou, and Dimitrios Kanoulas. Dipper: Diffusion-based 2d path planner applied on legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9264–9270. IEEE, 2024.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

David G Luenberger. *Optimization by vector space methods*. John Wiley & Sons, 1997.

Mitio Nagumo. Über die lage der integralkurven gewöhnlicher differentialgleichungen. *Proceedings of the physico-mathematical society of Japan. 3rd Series*, 24:551–559, 1942.

Ahmed H Qureshi, Anthony Simeonov, Mayur J Bency, and Michael C Yip. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2118–2124. IEEE, 2019.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020.

Mohit Srinivasan, Samuel Coogan, and Magnus Egerstedt. Control of multi-agent systems with finite time control barrier certificates and temporal logic. In *2018 IEEE Conference on Decision and Control (CDC)*, pp. 1991–1996. IEEE, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Kim P Wabersich, Andrew J Taylor, Jason J Choi, Koushil Sreenath, Claire J Tomlin, Aaron D Ames, and Melanie 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.

Kim Peter Wabersich and Melanie N Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597, 2021.

Li Wang, Aaron D Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3):661–674, 2017.

Xiao Wei, Wang Tsun-Hsuan, Gan Chuang, Hasani Ramin, Lechner Mathias, and Rus Daniela. Safediffuser: Safe planning with diffusion probabilistic models. IEEE, 2025.

Zebin Xing, Xingyu Zhang, Yang Hu, Bo Jiang, Tong He, Qian Zhang, Xiaoxiao Long, and Wei Yin. Goalflow: Goal-driven flow matching for multimodal trajectories generation in end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1602–1611, June 2025.

Brian Yang, Huangyuan Su, Nikolaos Gkanatsios, Tsung-Wei Ke, Ayush Jain, Jeff Schneider, and Katerina Fragkiadaki. Diffusion-es: Gradient-free planning with diffusion for autonomous and instruction-guided driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15342–15353, 2024.

Sean Ye and Matthew C Gombolay. Efficient trajectory forecasting and generation with conditional flow matching. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2816–2823. IEEE, 2024.

Fan Zhang and Michael Gienger. Robot manipulation with flow matching. In *CoRL 2024 Workshop on Mastering Robot Manipulation in a World of Abundant Data*, 2024.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.

A Proofs of Lemma 2 and Lemma 3

Proof of Lemma 2.
Let $\phi_t(\varepsilon) = \boldsymbol{\tau}_1 + \delta\varepsilon$, where $\delta \triangleq 1 - t$. We have push forward of p_ε under ϕ_t:

$$p_t(\boldsymbol{\tau} \mid \boldsymbol{\tau}_1) = [\phi_t]_* p_\varepsilon(\varepsilon) = p_\varepsilon(\phi_t^{-1}(\boldsymbol{\tau}))\; det\left[\frac{\partial \phi_t^{-1}}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau})\right] = \frac{1}{\delta^{d(H+1)}} p_\varepsilon\left(\frac{\boldsymbol{\tau} - \boldsymbol{\tau}_1}{\delta}\right)$$

By Bayes' rule,

$$p(\boldsymbol{\tau}_1 \mid \boldsymbol{\tau}_t^c) \;\propto\; p_1(\boldsymbol{\tau}_1)\, p_\varepsilon\left(\frac{\boldsymbol{\tau}_t^c - \boldsymbol{\tau}_1}{\delta}\right).$$

Since $-\log p_\varepsilon(z)$ is C^2 near 0 with Hessian $A \succ 0$ by the assumption,

$$-\log p_\varepsilon(z) \;=\; \tfrac{1}{2} z^\top A z + O(\|z\|^3).$$

Let $y = \boldsymbol{\tau}_1 - \boldsymbol{\tau}_t^c$. Substituting $z = y/\delta$ yields the posterior energy

$$\Phi_\delta(y) \;=\; \tfrac{1}{2\delta^2} y^\top A y - \log p_1(\boldsymbol{\tau}_t^c + y) + O(1).$$

The quadratic term dominates as $\delta \to 0\,(t \to 1)$, so the posterior concentrates in an $O(\delta)$ neighborhood of $\boldsymbol{\tau}_t^c$.

The stationarity condition $\nabla \Phi_\delta(y) = 0$ gives

$$\frac{1}{\delta^2} A y - \nabla \log p_1(\boldsymbol{\tau}_t^c + y) = 0.$$

Taylor expanding $\nabla \log p_1$ at $\boldsymbol{\tau}_t^c$ shows $y = O(\delta^2)$. Thus the posterior mode is

$$\hat{\boldsymbol{\tau}}_1 = \boldsymbol{\tau}_t^c + \delta^2 A^{-1} \nabla \log p_1(\boldsymbol{\tau}_t^c) + O(\delta^3).$$

Laplace's approximation then yields the same expansion for the posterior mean:

$$\mathbb{E}[\boldsymbol{\tau}_1 \mid \boldsymbol{\tau}_t^c] = \boldsymbol{\tau}_t^c + \delta^2 A^{-1} \nabla \log p_1(\boldsymbol{\tau}_t^c) + O(\delta^3).$$

Under the assumption, we have $\boldsymbol{\tau}_t^c = \boldsymbol{\tau}_1^\star + \delta\varepsilon$ with $\|\varepsilon\| = O(1)$,

$$\mathbb{E}[\boldsymbol{\tau}_1 \mid \boldsymbol{\tau}_t^c] = \boldsymbol{\tau}_1^\star + \delta\varepsilon + O(\delta^2) = \boldsymbol{\tau}_1^\star + O(\delta).$$

This proves Lemma 2.

Proof of Lemma 3.
If the flow dynamics follow the vanishing time-scaled flow dynamics (10), then we have:

$$\dot{\boldsymbol{\tau}}_t^c = \alpha(1-t)\, v_t(\boldsymbol{\tau}_t^c; \theta) = \alpha(\mathbb{E}[\boldsymbol{\tau}_1 \mid \boldsymbol{\tau}_t^c] - \boldsymbol{\tau}_t^c).$$

Let $\mathbf{e}_t \triangleq \boldsymbol{\tau}_t^c - \boldsymbol{\tau}_1^\star \in \mathbb{R}^{d\times(H+1)}$, and denote its k-th column by $e_{k,t} \in \mathbb{R}^d$. By Lemma 2, $\mathbb{E}[\boldsymbol{\tau}_1 \mid \boldsymbol{\tau}_t^c] = \boldsymbol{\tau}_1^\star + O(1-t)$ as $t \to 1$, hence we have

$$\dot{e}_{k,t} = -\alpha e_{k,t} + O(1-t).$$

Solving with an integrating factor gives

$$e_{k,t} = e^{-\alpha t} e_{k,0} + \alpha e^{-\alpha t} \int_0^t e^{\alpha s} O(1-s)\, ds = (e_{k,0} + O(1))e^{-\alpha t} + O((1-t)^2).$$

Combining the column vectors again yields the form

$$\mathbf{e}_t = (\mathbf{e}_0 + O(1))e^{-\alpha t} + O\big((1-t)^2\big), \qquad \mathbf{e}_0 = \varepsilon.$$

which proves Lemma 3.

B Proof of Theorem 1 and Proposition 1

We drop the superscript c for simplicity, and choose the Lyapunov candidate function $V(\boldsymbol{\tau}_t^k) \triangleq \max(\delta - b(\boldsymbol{\tau}_t^k), 0)$. Since $w(t) = 0$ for all $t \geq t_w$, the barrier inequality (14) reduces on $[t_w, 1]$ to

$$\dot{b}(\boldsymbol{\tau}_t^k) + \epsilon \cdot \operatorname{sgn}\big(b(\boldsymbol{\tau}_t^k) - \delta\big)\, \big|b(\boldsymbol{\tau}_t^k) - \delta\big|^\rho \;\geq\; 0.$$

Case 1: If $\boldsymbol{\tau}_{t_w}^k \in \mathcal{C}_\delta$ (i.e., $b(\boldsymbol{\tau}_{t_w}^k) \geq \delta$), then $V(\boldsymbol{\tau}_{t_w}^k) = 0$. For all $t \geq t_w$, if $b(\boldsymbol{\tau}_t^k) > \delta$ we have $V(\boldsymbol{\tau}_t^k) = 0$. If $b(\boldsymbol{\tau}_t^k) = \delta$, the barrier inequality (14) with $\operatorname{sgn}(0) = 0$ reduces to $\dot{b}(\boldsymbol{\tau}_t^k) \geq 0$, so the path cannot exit $\mathcal{C}_\delta$ by Nagumo's principle (Nagumo, 1942)[4]. Therefore $V(\boldsymbol{\tau}_t^k) = 0$ for all $t \geq t_w$, which implies $\boldsymbol{\tau}_t^k \in \mathcal{C}_\delta$; the system stays in $\mathcal{C}_\delta$.

Case 2: If $\boldsymbol{\tau}_{t_w}^k \notin \mathcal{C}_\delta$ (i.e., $b(\boldsymbol{\tau}_{t_w}^k) < \delta$), then $V(\boldsymbol{\tau}_t^k) = \delta - b(\boldsymbol{\tau}_t^k) > 0$. The following finite-stability condition holds

$$\dot{V}(\boldsymbol{\tau}_t^k) = -\dot{b}(\boldsymbol{\tau}_t^k) \leq -\epsilon(\delta - b(\boldsymbol{\tau}_t^k))^\rho = -\epsilon V(\boldsymbol{\tau}_t^k)^\rho.$$

Define the comparison system

$$\dot{\phi}(t) = -\epsilon\phi(t)^\rho,\ \phi(t_w) = V(\boldsymbol{\tau}_{t_w}^k).$$

By the Comparison Lemma (See Lemma 3.4 in Khalil & Grizzle (2002)), we have:

$$V(\boldsymbol{\tau}_t^k) \leq \phi(t),\ \forall t \geq t_w.$$

The solution $\phi(t)$ is

$$\phi(t) = \big(V(\boldsymbol{\tau}_{t_w}^k)^{1-\rho} - (1-\rho)\epsilon(t - t_w)\big)^{\frac{1}{1-\rho}},\ \text{for } t \geq t_w.$$

Thus,

$$V(\boldsymbol{\tau}_t^k) \leq \big(V(\boldsymbol{\tau}_{t_w}^k)^{1-\rho} - (1-\rho)\epsilon(t - t_w)\big)^{\frac{1}{1-\rho}}.$$

Hence, the state reaches the robust safe set $\mathcal{C}_\delta$ in finite time T that satisfies $V(\boldsymbol{\tau}_t^k) \leq \phi(T) = 0$. Moreover, we get the finite convergence time,

$$T = t_w + \frac{V(\boldsymbol{\tau}_{t_w}^k)^{1-\rho}}{\epsilon(1-\rho)} = t_w + \frac{(\delta - b(\boldsymbol{\tau}_{t_w}^k))^{1-\rho}}{\epsilon(1-\rho)}.$$

Therefore, for all $t \geq T$, we have $V(\boldsymbol{\tau}_t^k) \leq 0$, implying $\mathbf{x} \in \mathcal{C}_\delta$. This completes the proofs of both Theorem 1 and Proposition 1.

[4]Nagumo's theorem states that if the vector field at the boundary lies in the tangent cone of a set, then the set is forward invariant.

C Differences in Local Trap Definitions

We clarify the difference between the local trap definition used in our SafeFlowMatcher and that of the baseline method SafeDiffuser (Wei et al., 2025).

Definition 4 (Local Trap in SafeDiffuser) *A local trap problem occurs during the planning process if there exists $k \in \mathcal{H}$ such that $b(\boldsymbol{\tau}_1^k) = 0$ and $\|\boldsymbol{\tau}_1^k - \boldsymbol{\tau}_1^{k-1}\| > \zeta$, where $\zeta > 0$ is a user-defined threshold depending on the planning environment.*

In contrast, our definition of a local trap in SafeFlowMatcher removes the condition $b(\boldsymbol{\tau}_1^k) = 0$ and instead considers only the abrupt discontinuity in the path. The reason for relaxing the condition is illustrated in Figure 8. In this example, the generated path is incomplete due to overly strong or early intervention of the CBF. However, since the waypoints do not strictly lie on the boundary (i.e., $b(\boldsymbol{\tau}_1^k) \neq 0$), the original SafeDiffuser definition fails to detect this failure as a local trap. Therefore, we generalize the definition to capture a wider class of failure cases.

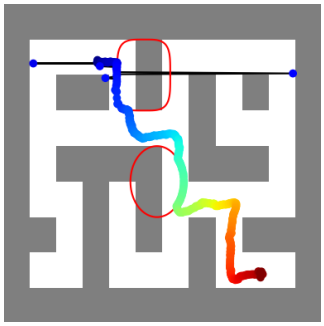

Figure 8: Although some waypoints do not violate constraints (i.e., $b(\boldsymbol{\tau}_t^k) > 0$), it fails to reach the goal. Our definition considers such cases as local traps, while the original definition does not.

D Experimental Details

D.1 Experimental Setup

All CBF constraints are enforced via the closed-form projection of the CBF-QP in (16). For each model family, the safety-enabled variant (SafeDiffuser/ SafeDDIM or SafeFM/SafeFlowMatcher) reuses the same trained weights as its non-safe counterpart. We match the total number of samples seen during training across methods, *batch_size* $\times$ *#iterations* $= 6.4 \times 10^7$. Unless specified, the default number of function evaluations (NFE) at inference time is 256.

Maze2D We first swept through several batch sizes to ensure that the number of data checked during the training process was 6.4×10^7, the same amount of data as used to train Diffuser Janner et al. (2022). Our results are shown in Table 5; both Diffuser and FM performed best or on par at 128. Thus, for all models, including SafeDiffuser and its variants, we used batch size 128 for all our Maze2D experiments. Other training and inference hyperparameters are shown in Tables 6 and 7.

Table 5: Scores by Batch Size for Maze2D for both Diffuser and FM.

	Batch size	16	32	64	128	256
Score(↑)	**FlowMatcher**	1.631 ± 0.003	1.628 ± 0.002	1.615 ± 0.031	1.631 ± 0.003	1.523 ± 0.196
	Diffuser (Janner et al., 2022)	1.503 ± 0.424	1.438 ± 0.500	1.516 ± 0.316	1.537 ± 1.537	1.536 ± 0.338

Locomotion. Extending the insights gained from Maze2D, locomotion was also trained using the same batch size of 128. SafeFlowMatcher and SafeFM shares the same hyperparameter values and conditions as shown in Table 8.

We keep the total number of seen samples constant during batch-size sweeps: *batch_size* $\times$ *#iterations* $= 6.4 \times 10^7$. Both FlowMatcher and Diffuser achieved the best validation score at *batch_size* $= 128$ also, which we adopt throughout Maze2D and Locomotion tasks. Unless otherwise stated, the number of function evaluations (NFE) during inference equals the number of diffusion steps, 256.

Table 6: Maze2D's training hyper parameters

Train	
Loss type	`L2`
Training steps n_{train}	5.0×10^5
Steps per epoch	2500
Batch size	128
Learning rate	3×10^{-4}
Gradient accumulate every	1
EMA decay	0.995

Table 7: Maze2D's evaluation hyper parameters

Evaluation	
Horizon H	256
Diffusion steps T	256
NFE	256 (by default; equals T)

Table 8: Locomotion (Walker2d/Hopper)'s training hyperparameters

Train	
Loss type	`l2`
Training steps n_{train}	2.5×10^5
Steps per epoch	2500
Batch size	128
Learning rate	2×10^{-4}
Gradient accumulate every	1
EMA decay	0.995

Table 9: Locomotion value network's training hyperparameters

Train Value	
Loss type	`L2`
Training steps n_{train}	5.0×10^4
Steps per epoch	2500
Batch size	128
Learning rate	2×10^{-4}
Gradient accumulate every	1
EMA decay	0.995

Table 10: Locomotion evaluation hyper parameters

Hyperparameter	Value
Horizon H	600
Diffusion steps T	20
NFE	20

D.2 Performance Metrics

BS measures safety constraint satisfaction using CBFs for each safety constraint in the environment. Specifically, these metrics compute the minimum value of the barrier function $b(\boldsymbol{\tau}_t^k)$ across some total number N of test runs

$$\min_{i=1,2,\ldots,N} \min_{k \in \mathcal{H}} b(\boldsymbol{\tau}_T^k).$$

There are two safety constraints in our Maze2D experiment. Their barrier functions are defined as

$$\textbf{BS1}: \left(\frac{x-x_0}{a}\right)^2 + \left(\frac{y-y_0}{b}\right)^2 \geq 1,$$

$$\textbf{BS2}: \left(\frac{x-x_0}{a}\right)^4 + \left(\frac{y-y_0}{b}\right)^4 \geq 1,$$

where $(x, y) \in \mathbb{R}^2$ denotes the agent's 2D state, $(x_0, y_0) \in \mathbb{R}^2$ specifies the center of the obstacle, and $a, b > 0$ are scaling parameters that shape the corresponding safety region.

The barrier function for our Locomotion tasks is defined as $z + \varphi\, v_z \leq h_r$, where $h_r > 0$ denotes the roof height, $\varphi > 0$ is a scaling parameter, and $v_z \in \mathbb{R}$ is the robot head's vertical velocity. This speed-dependent constraint ensures that the head position z plus its velocity contribution remains below the roof limit.

Score is a normalized, undiscounted performance metric that reflects task success. For Maze2D, episodes have horizon $H{=}1000$; once the agent enters a goal neighborhood, it receives unit reward for each remaining step, so the score equals the fraction of remaining steps and lies in $[0, 1]$. For locomotion tasks, the score is proportional to forward displacement and is normalized so that reaching the target position $x{=}1$ yields a score of 1.

Trap Rate measures the rate of local traps, i.e., the percentage of episodes in which the generated path becomes stuck against barrier constraints; see Definition 2.

Time reports the average wall-clock time per sampling step. We use a uniform time discretization with $\Delta t = 1/T$ for a total of T steps. For each method, we divide the *total* sampling time (including all CBF/QP overheads) by the number of integration updates it performs $\mathrm{Time} = \frac{\text{Total sampling time (incl. CBF)}}{\#\text{ of sampling steps}}$.

- For diffusion-based baselines (e.g., SafeDiffuser) with $T{=}256$, the per-step time is $\mathrm{Time} = \mathrm{Total}/256$.
- For SafeFlowMatcher (FM with prediction–correction), if we use T^p prediction steps and T^c correction steps (e.g., $T^p{=}1$, $T^c{=}256$), then $\mathrm{Time} = \frac{\mathrm{Total}}{T^p + T^c} = \frac{\mathrm{Total}}{257}$.

Curvature ($\boldsymbol{\kappa}$) measures the directional change of a path. For each discrete segment, we compute the angular deviation relative to the previous step. The metric is defined as the average curvature along the path.

Acceleration ($\boldsymbol{\alpha}$) captures the change in velocity across consecutive time steps. It is computed as the mean squared acceleration magnitude along the path.