[TINY PAPER] SAFE STREAMING FLOW PLANNING BY ALIGNING GENERATION WITH EXECUTION

Seunghwan Jang[1]*, **Jeongyong Yang**[1]*, **Siddharth Ancha**[2], **SooJean Han**[1]
[1]Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea
[2]University of California, Berkeley, CA, USA
{jsh991124, seiryu2238, soojean}@kaist.ac.kr
sancha@berkeley.edu

ABSTRACT

Generative planners based on diffusion/flow-matching can learn to synthesize long-horizon trajectories from demonstrations. However, real-world deployment requires (i) certified collision avoidance and (ii) tight online replanning at fast execution rates. Prior safe diffusion/flow planners generate the agent's full trajectory *at once*, while repeatedly projecting intermediate noisy trajectories to satisfy barrier constraints. This approach is not only computationally intensive, but also introduces distribution shift since the learned sampling dynamics is distinct from the system's execution dynamics. We propose `SafeStreamingFlow`, a goal-conditioned planner that aligns flow sampling dynamics with execution dynamics by sequentially integrating a learned state-space vector field. Importantly, we need to certify safety *only for the executed step* via discrete-time high order control barrier functions. To improve robustness, we introduce a hierarchical network architecture that predicts the next position before predicting the next physical velocity. On D4RL Maze2D, `SafeStreamingFlow` reduces planning latency and improves execution-time safety compared to prior methods, while maintaining competitive goal-reaching success.

1 INTRODUCTION

Diffusion/flow based generative planners have recently attracted significant attention for their ability to learn long-horizon, multi-modal behaviors directly from demonstration data (Janner et al., 2022; Carvalho et al., 2023; Lipman et al., 2023). Despite these advantages, their deployment on robotic systems remains challenging. In particular, online planning of safe trajectories is largely unresolved.

While guidance-based methods can guide generative planners toward safer behavior, they do not provide formal safety guarantees during execution (Mizuta & Leung, 2024). Control barrier functions (CBFs) are a standard tool for guaranteeing safety in planning and control (Ames et al., 2019). Conventional diffusion/flow methods generate the entire trajectory at once — they start from random noise and gradually denoise the trajectory to simultaneously predict all states over the task horizon. However, trajectory-level diffusion is ill-suited for CBFs. Current methods must apply CBFs to correct intermediate *noisy* trajectory samples (Xiao et al., 2025). Such corrections act on the model's denoising dynamics rather than the physical system's dynamics — safety is enforced in the diffusion/flow denoising time, while the system state continues to evolve in physical time. This mismatch can break dynamics consistency, *i.e.* the alignment between generated state transitions and the system's execution dynamics, leading to unreliable rollouts. Furthermore, online planning is slow since online planning requires regenerating a whole trajectory at every execution cycle.

To address these challenges, we adopt a streaming formulation of flow matching (Jiang et al., 2025) in order to align generation, execution, and safety certification in time. We propose `SafeStreamingFlow`, which generates the next state in each execution cycle with formal safe guarantees. This yields an execution-aligned generative model that is suitable for online planning, incorporates real-time feedback and is certifiably safe.

*Equal contribution.

Contributions: (i) We propose a new generative planning method using streaming flow with state-space vector fields integrated in *physical* time. (ii) We propose a hierarchical state prediction architecture that masks velocity inputs to next-position prediction and improves robustness under safety filtering. (iii) We introduce a safe planning framework that certifies the executed step via discrete-time HOCBF-QP (Xiong et al., 2023), enabling safe and efficient closed-loop planning.

2 Problem Formulation

We consider a control-affine dynamical system of the form

$$\dot{\mathbf{x}}_t = f(\mathbf{x}_t) + g(\mathbf{x}_t)\mathbf{u}_t = \begin{bmatrix} \mathbf{v}_t \\ \tilde{f}(\mathbf{p}_t, \mathbf{v}_t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \tilde{g}(\mathbf{p}_t, \mathbf{v}_t) \end{bmatrix} \mathbf{u}_t, \tag{1}$$

where $\mathbf{x}_t = (\mathbf{p}_t, \mathbf{v}_t) \in \mathcal{X} \subset \mathbb{R}^{2n}$ denotes the system state ($\mathcal{X}$ is the feasible state space), $\mathbf{p}_t \in \mathbb{R}^n$ and $\mathbf{v}_t \in \mathbb{R}^n$ are position and velocity, respectively, and $\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^n$ is the control input ($\mathcal{U}$ is the admissible control input set). The functions $f : \mathcal{X} \to \mathbb{R}^{2n}$ and $g : \mathcal{X} \to \mathbb{R}^{2n\times n}$ denote the full state drift and control vector fields, respectively. We use $\tilde{f} : \mathcal{X} \to \mathbb{R}^n$ and $\tilde{g} : \mathcal{X} \to \mathbb{R}^{n\times n}$ to denote their acceleration-level components. All functions are assumed to be locally Lipschitz continuous. This formulation captures a broad class of second-order systems, including point-mass and mass–spring–damper dynamics, and is widely used as a tractable approximation of more complex robotic and mechanical systems, where the control input acts directly on the acceleration dynamics (i.e., $\tilde{g}(\mathbf{p}, \mathbf{v})$ has full column rank).

Let $T \in \mathbb{R}_{\geq 0}$ denote the planning horizon. A trajectory is defined as $\boldsymbol{\tau} : [0, T] \to \mathcal{X}$ that satisfies system dynamics (1), i.e., $\boldsymbol{\tau} = \{\mathbf{x}_t \mid t \in [0, T]\}$. We are given a dataset of N expert demonstrations $\mathcal{D} = \{\boldsymbol{\tau}^{(k)}\}_{k=1}^N$, where each demonstration starts at $\mathbf{x}_0^{(k)}$ and terminates at a goal state $\mathbf{x}_T^{(k)}$.

Safety constraints are specified by continuously differentiable barrier functions $\{b_i : \mathcal{X} \to \mathbb{R}\}_{i\in\mathcal{I}}$, where $\mathcal{I}$ is a finite index set corresponding to obstacles. The safe set is defined as the intersection

$$\mathcal{C} \triangleq \bigcap_{i\in\mathcal{I}} \{\mathbf{x} \in \mathcal{X} : b_i(\mathbf{x}) \geq 0\} \tag{2}$$

Problem 1 (Safe planning from demonstrations) *Given the system dynamics (1), demonstration dataset $\mathcal{D}$, start state $\mathbf{x}_s$, goal state $\mathbf{x}_g$, planning horizon T, and a safe set $\mathcal{C}$ defined in (2), design a planner that generates a state trajectory $\boldsymbol{\tau}(t)$ such that $\mathbf{x}_0 = \mathbf{x}_s$, $\mathbf{x}_T = \mathbf{x}_g$, and $\mathbf{x}_t \in \mathcal{C}$ for all $t \in [0, T]$, and $\boldsymbol{\tau}$ satisfies the system dynamics (1).*

3 Method

Our approach consists of two components: (i) streaming generation of state trajectories using a learned (start, goal)-conditioned vector field, and (ii) safety certification applied only to the executed control input.

3.1 Streaming Flow Planning

We formulate global planning as forward integration of a learned, time-dependent vector field. Given start state $\mathbf{x}_s$ and goal state $\mathbf{x}_g$, we model the state evolution using the following flow dynamics

$$\dot{\mathbf{x}}_t = F_\theta\big(\mathbf{x}_t, t \mid \mathbf{x}_s, \mathbf{x}_g\big), \quad t \in [0, T], \tag{3}$$

where F_θ is a neural vector field. Nominal trajectories are generated by integrating (3) forward over the generation time $t \in [0, T]$[1]. Here, we align the generation (sampling) time with planning time.

Unlike non-sequential diffusion/flow planners that generate the entire horizon at once, our formulation produces sequential one-step state updates. During execution, the vector field is conditioned on the

[1]Most flow matching formulations parameterize time by a normalized variable $s \in [0, 1]$. Instead, we treat flow generation time as identical to physical time $t \in [0, T]$, where T is the planning horizon. This is an equivalent re-parameterization corresponding to $s = t/T$, which yields $\frac{d\boldsymbol{\tau}}{dt} = \frac{1}{T}\frac{d\boldsymbol{\tau}}{ds}$.

current state at every execution cycle and used to generate only a one-step-ahead reference. This yields a streaming planner whose generation is aligned with execution in physical time.

The vector field F_θ is trained using conditional flow matching on expert state trajectories. Given a demonstration $\boldsymbol{\tau}(t)$ with start–goal pair $(\mathbf{x}_s, \mathbf{x}_g)$, we construct a stabilizing target vector field $F_{\boldsymbol{\tau}}(\mathbf{x}, t) \triangleq \dot{\boldsymbol{\tau}}(t) - k(\mathbf{x} - \boldsymbol{\tau}(t))$, with the initial distribution $p^0_{\boldsymbol{\tau}}(\mathbf{x}) = \mathcal{N}\big(\mathbf{x} \mid \boldsymbol{\tau}(0), \sigma_0^2 I\big)$ where $k \geq 0$ is a stabilization gain and σ_0 is a small initial standard deviation (Jiang et al., 2025). This field induces a thin Gaussian tube $p_\tau(\mathbf{x} \mid t) = \mathcal{N}\big(\mathbf{x} \mid \boldsymbol{\tau}(t), \sigma_t^2 I\big)$ of trajectories centered around $\boldsymbol{\tau}(t)$ as shown in Jiang et al. (2025). We train F_θ by minimizing the mean squared error

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{\tau}\sim\mathcal{D}}\, \mathbb{E}_{t\sim\mathrm{Uniform}(0,T)}\, \mathbb{E}_{\mathbf{x}\sim p_\tau(\mathbf{x}|t)} \big\| F_\theta(\mathbf{x}, t \mid \boldsymbol{\tau}(0), \boldsymbol{\tau}(T)) - F_{\boldsymbol{\tau}}(\mathbf{x}, t) \big\|_2^2 \tag{4}$$

The vector field learned by optimizing this objective is known to match the per-time conditional state marginals induced by the demonstrations. As in standard conditional flow matching, it does not necessarily enforce matching of the joint trajectory distribution.

Although the vector field F_θ is defined in continuous time, in practice, planning and execution are carried out in discrete time. We discretize the planning time $[0, T]$ into H uniform planning horizon steps with step size $\Delta t = T/H$, and define the discrete planning time set as $\mathcal{T} \triangleq \{0, \Delta t, 2\Delta t, \ldots, (H-1)\Delta t\}$. Accordingly, we propagate the flow dynamics (3) as follows[2]:

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t \cdot F_\theta(\mathbf{x}_t, t \mid \mathbf{x}_s, \mathbf{x}_g), \qquad t \in \mathcal{T} \tag{5}$$

Hierarchical state prediction (HSP): In a second-order system, a naive approach is to predict the full state $(\hat{\mathbf{p}}_{t+\Delta t}, \hat{\mathbf{v}}_{t+\Delta t})$ from the full current state $(\mathbf{p}_t, \mathbf{v}_t)$ using the MSE loss in (4). However, this can lead to degenerate solutions that do not require learning goal-directed planning behavior.

Specifically, the MSE implicitly targets the conditional expectation $\mathbb{E}\big[\mathbf{p}_{t+\Delta t} \mid \mathbf{x}_t, \mathbf{x}_s, \mathbf{x}_g\big]$. Due to the kinematic relationship $\dot{\mathbf{p}} = \mathbf{v}$ ($\mathbf{p}_{t+\Delta t} \approx \mathbf{p}_t + \mathbf{v}_t\, \Delta t$), the model can minimize the training loss by directly exploiting the current velocity $\mathbf{v}_t$, without relying on goal conditioning. This *kinematic shortcut* weakens global planning behavior, making the planner sensitive to velocity perturbations from tracking error or safety filtering.

To address this problem, we introduce a hierarchical state prediction that factorizes F_θ into a position branch F^p_θ and a velocity branch F^v_θ. Specifically, the position branch predicts next position *without* conditioning on $\mathbf{v}_t$ in order to prevent the kinematic shortcut:

$$\textit{(Step 1)} \quad \dot{\mathbf{p}}_t = F^p_\theta(\mathbf{p}_t, t \mid \mathbf{x}_s, \mathbf{x}_g) \qquad \textit{(Step 2)} \quad \hat{\mathbf{p}}_{t+\Delta t} = \mathbf{p}_t + \dot{\mathbf{p}}_t\, \Delta t \tag{6}$$

$$\textit{(Step 3)} \quad \dot{\hat{\mathbf{v}}}_t = F^v_\theta(\mathbf{p}_t, \mathbf{v}_t, \hat{\mathbf{p}}_{t+\Delta t}, t \mid \mathbf{x}_s, \mathbf{x}_g) \qquad \textit{(Step 4)} \quad \hat{\mathbf{v}}_{t+\Delta t} = \mathbf{v}_t + \dot{\hat{\mathbf{v}}}_t\, \Delta t \tag{7}$$

In HSP, F^p_θ is conditioned on $(\mathbf{x}_s, \mathbf{x}_g)$ per step, while F^v_θ is conditioned on $\hat{\mathbf{p}}_{t+\Delta t}$. This conditioning mitigates numerical errors better than standard numerical integrators, while preserving kinematic consistency overall. This property of HSP is further studied via the ablation study in Appendix B.

3.2 Safe Streaming Flow Planning via Control Barrier Functions

We couple streaming state generation with PD control and safety filtering to form a complete planning and execution loop. At each execution cycle, the generative model produces a one-step-ahead state reference $(\hat{\mathbf{p}}_{t+\Delta t}, \hat{\mathbf{v}}_{t+\Delta t})$ via (6)–(7). This reference is tracked by a PD controller. Specifically, the nominal control input is given by

$$\mathbf{u}_t^{\mathrm{PD}} = K_p(\hat{\mathbf{p}}_{t+\Delta t} - \mathbf{p}_t) + K_v(\hat{\mathbf{v}}_{t+\Delta t} - \mathbf{v}_t) \tag{8}$$

where K_p and K_v are proportional and derivative gains, respectively.

Since the system (1) is executed in discrete time with step size Δt, we adopt discrete-time high order control barrier functions (HOCBFs) (Xiong et al., 2023) to certify safety directly at the discrete level, avoiding the sampled-data mismatch of continuous-time formulations. For each barrier function b_i with relative degree two, we define the barrier sequence $\psi_{i,0}(\mathbf{x}_k) \triangleq b_i(\mathbf{x}_k)$, $\psi_{i,1}(\mathbf{x}_k) \triangleq \psi_{i,0}(\mathbf{x}_{k+1}) - \psi_{i,0}(\mathbf{x}_k) + \alpha_{i,1}(\psi_{i,0}(\mathbf{x}_k))$, and enforce the HOCBF condition:

$$\psi_{i,1}(\mathbf{x}_{k+1}) - \psi_{i,1}(\mathbf{x}_k) + \alpha_{i,2}(\psi_{i,1}(\mathbf{x}_k)) \geq 0, \quad \forall i \in \mathcal{I}, \tag{9}$$

[2]This corresponds to a forward Euler integration; higher-order numerical integrators can also be used.

where $\alpha_{i,1}, \alpha_{i,2}$ are class-$\mathcal{K}$ functions. Since (9) is affine in $\mathbf{u}$, we compute the safe control by solving:

$$\mathbf{u}_t^{\text{safe}} = \underset{\mathbf{u}_t \in \mathcal{U}}{\operatorname{argmin}} \|\mathbf{u}_t - \mathbf{u}_t^{\text{PD}}\|_2^2 \quad \text{subject to (9)} \tag{10}$$

which minimally deviates from the PD input while guaranteeing forward invariance of the safe set in discrete time.

Finally, we propagate the known system dynamics (1) by using $\mathbf{u}_t^{\text{safe}}$ to obtain the next state $\mathbf{x}_{t+\Delta t}^{\text{safe}}$. This safe state is then fed back into the velocity field for the next streaming update. We summarize the planning procedure in Appendix A.

4 Experiments

We evaluate on D4RL Maze2D (Fu et al., 2020) and compare our methods StreamingFlow and `SafeStreamingFlow` against Diffuser (Janner et al., 2022) and SafeDiffuser (Xiao et al., 2025). The "Safe" prefix indicates whether we apply the safety filter described in Section 3.2. Full implementation details are provided in Appendix D.

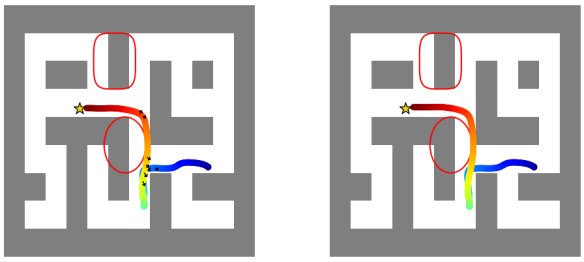

Figure 1: Plan (left) and rollout (right). $\star$ represents the goal $\mathbf{x}_g$. Color line represents the trajectory $\boldsymbol{\tau}(t)$ from the start (blue) to the end (red). Black arrows visualize stepwise HOCBF-QP safety corrections.

We report success rate, plan safety (fraction of safe plans), rollout safety (fraction of rollouts with zero barrier violations), and total planning time (inference + QP). We fix H=384 for all methods; diffusion baselines use K=256 denoising steps, while our streaming model uses K=384 steps matching the planning horizon. We focus on safe variants, for which planning time includes both model inference and CBF/HOCBF-QP computation.

While trajectory-level safe planners require $H \times K$ QP calls, `SafeStreamingFlow` requires only H—one per execution step—achieving 2.5–3$\times$ lower latency than all safe baselines with substantially higher rollout safety (Table 1). The higher nominal wall-clock time of StreamingFlow is due to sequential in-place operations rather than model size; see Appendix E for a detailed computational analysis. See Appendix C for theoretical support that sequential generation inherits goal-reaching from the conditional structure.

Table 1: Maze2D Large results: success rate, plan/roll-out safety rate, and total plan time.

Method	Success (%)	Plan Safety (%)	Rollout Safety (%)	Plan Time (s)
Diffuser (Janner et al., 2022)	70.3	–	–	1.052
FM (Yang et al., 2026)	63.3	–	–	1.009
FlowMatcher (Yang et al., 2026)	64.8	–	–	1.015
StreamingFlow (open-loop)	**89.7**	–	–	1.865
StreamingFlow (closed-loop)	88.6	–	–	2.163
SafeDiffuser (Xiao et al., 2025)	66.9	100	38.0	10.227
SafeFM (Yang et al., 2026)	53.6	100	43.7	10.245
SafeFlowMatcher (Yang et al., 2026)	65.2	100	53.8	10.267
`SafeStreamingFlow` (open-loop)	84	100	81.0	3.536
`SafeStreamingFlow` (closed-loop)	**88**	100	**96.0**	4.001

5 Conclusion

We presented `SafeStreamingFlow`, a streaming global planner that aligns generation with execution and certifies safety via an HOCBF-QP at each executed step. By generating only the next reference state and propagating safety corrections forward through known dynamics, `SafeStreamingFlow` supports tight online replanning without additional computational overhead. On Maze2D, it improves rollout safety and reduces planning latency compared to trajectory-level safe planners. The safety margin δ is currently set heuristically; principled robust bounds and evaluation on higher-dimensional robotic tasks are immediate future directions.

REFERENCES

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.

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.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Siddharth Ancha. Streaming flow policy: Simplifying diffusion/flow-matching policies by treating action trajectories as flow trajectories. In *9th Annual Conference on Robot Learning*, 2025.

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.

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

Wei Xiao, Tsun-Hsuan Wang, Chuang Gan, Ramin Hasani, Mathias Lechner, and Daniela Rus. SafeDiffuser: Safe planning with diffusion probabilistic models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yuhan Xiong, Di-Hua Zhai, Mahdi Tavakoli, and Yuanqing Xia. Discrete-time control barrier function: High-order case and adaptive case. *IEEE Transactions on Cybernetics*, 53(5):3231–3239, 2023.

Jeongyong Yang, Seunghwan Jang, and SooJean Han. Safeflowmatcher: Safe and fast planning using flow matching with control barrier functions. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=refcXHU1Nh.

A PSEUDOCODE FOR SAFESTREAMINGFLOW

Here, we provide the pseudocode of our proposed method.

Algorithm 1 SafeStreamingFlow

Require: Dynamics f and g; learned velocity field F_θ; start $\mathbf{x}_s$ and goal $\mathbf{x}_g$; planning time set $\mathcal{T}$; PD gains K_p and K_v; barrier functions $\{b_i\}_{i\in\mathcal{I}}$ with $\{\alpha_{i,0}, \alpha_{i,1}\}_{i\in\mathcal{I}}$; planning mode $\Delta \in \{\text{OPEN}, \text{CLOSED}\}$

Initialize $\mathbf{x}_0 = (\mathbf{p}_0, \mathbf{v}_0)$
for $t \in \mathcal{T}$ **do**
  if Δ == CLOSED **then**
    Observe $\mathbf{x}_t = (\mathbf{p}_t, \mathbf{v}_t)$
  end if
  Compute $(\hat{\mathbf{p}}_{t+\Delta t}, \hat{\mathbf{v}}_{t+\Delta t})$ via (6)–(7)
  $\mathbf{u}_t^{\text{PD}} \leftarrow K_p(\hat{\mathbf{p}}_{t+\Delta t} - \mathbf{p}_t) + K_v(\hat{\mathbf{v}}_{t+\Delta t} - \mathbf{v}_t)$ via (8)
  $\mathbf{u}_t^{\text{safe}} \leftarrow$ solve HOCBF-QP in (10)
  $\mathbf{x}_{t+\Delta t}^{\text{safe}} \leftarrow \mathbf{x}_t + \Delta t(f(\mathbf{x}_t) + g(\mathbf{x}_t)\mathbf{u}_t^{\text{safe}})$
  $\mathbf{x}_{t+\Delta t} \leftarrow \mathbf{x}_{t+\Delta t}^{\text{safe}}$
end for
return $\{\mathbf{x}_t\}_{t\in\mathcal{T}}$ and $\{\mathbf{u}_t^{\text{safe}}\}_{t\in\mathcal{T}}$.

B HIERARCHICAL STATE PREDICTION ABLATION: AVOIDING THE KINEMATIC SHORTCUT

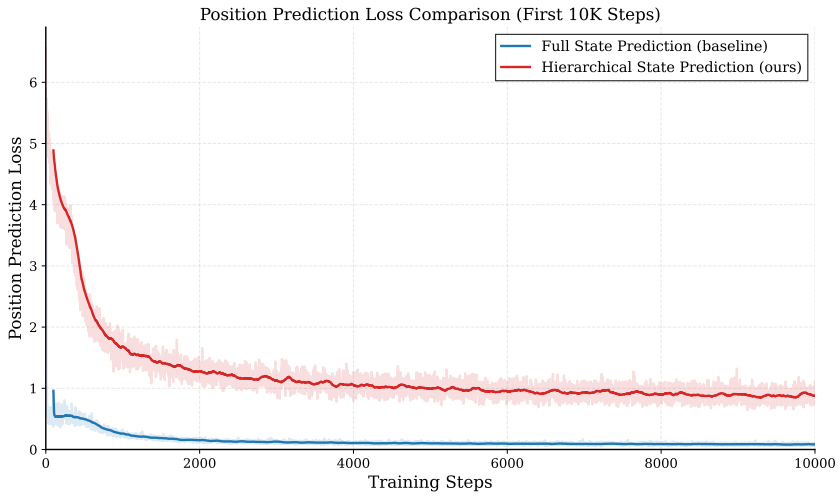

Figure 2: Position prediction loss during training. The FSP quickly reaches near-zero loss by exploiting the kinematic shortcut, whereas HSP converges more slowly to a higher plateau.

We ablate the proposed HSP, described in Section 3.2, against a full state prediction (FSP) that directly predicts $(\hat{\mathbf{p}}_{t+\Delta t}, \hat{\mathbf{v}}_{t+\Delta t})$ from the full current state $(\mathbf{p}_t, \mathbf{v}_t)$.

Figure 2 plots the position prediction loss $\|\hat{\mathbf{p}}_{t+\Delta t} - \mathbf{p}^\star_{t+\Delta t}\|_2^2$ during training, where $\hat{\mathbf{p}}_{t+\Delta t}$ denotes the predicted next position produced by the learned vector field F_θ and $\mathbf{p}^\star_{t+\Delta t}$ denotes the ground-truth position from the demonstration dataset $\mathcal{D}$. The HSP decreases slowly and saturates around 0.8, while the FSP quickly converges to near-zero loss, approximately 0.06.

This gap does *not* indicate better planning. The FSP can minimize the loss by exploiting what we refer to as a kinematic shortcut. As a result, the next position is predicted almost deterministically from the current velocity, with little reliance on start–goal conditioning.

Table 2 reports the goal-reaching success rate under the same execution protocol. Because success requires reaching within a tight tolerance of the goal position, small accumulated errors can lead to failure. Despite its lower position loss, the FSP suffers error accumulation from safety interventions from (10), failing to reach the goal precisely in both open-loop and closed-loop settings. In contrast, HSP remains robust to such perturbations and achieves substantially higher success rate in both settings.

Table 2: Goal-reaching success rate (%) in open-loop and closed-loop execution. Lower position prediction loss does not necessarily imply better execution when the loss can be minimized by a kinematic shortcut.

	Success Rate (%)	
Method	Open-loop	Closed-loop
Full State Prediction	39.6	5.8
Hierarchical State Prediction (ours)	**89.7**	**88.6**

C Why Goal Conditioning Encourages Goal-Reaching

This appendix provides intuition for why conditioning on the start–goal pair $(\mathbf{x}_s, \mathbf{x}_g)$ induces goal-reaching behavior. We interpret the learned vector field as approximating a Bayesian marginalization over goal-satisfying trajectories, thereby inheriting the terminal constraints of the demonstrations. This interpretation is idealized; empirical validation is provided in Section 4.

Goal-Reaching Support. The demonstration distribution $p_{\text{data}}(\boldsymbol{\tau} \mid \mathbf{x}_s, \mathbf{x}_g)$ is strictly supported on the manifold of trajectories that satisfy the boundary conditions:

$$\operatorname{supp}\big(p_{\text{data}}(\boldsymbol{\tau} \mid \mathbf{x}_s, \mathbf{x}_g)\big) \subseteq \mathcal{T}_{\mathbf{x}_s,\mathbf{x}_g} := \{\boldsymbol{\tau} : \boldsymbol{\tau}(0) = \mathbf{x}_s,\ \boldsymbol{\tau}(T) = \mathbf{x}_g\}. \tag{11}$$

The flow matching objective minimizes the MSE between the learned field F_θ and the target field $F_{\boldsymbol{\tau}}(\mathbf{x}, t) = \dot{\boldsymbol{\tau}}(t) - k(\mathbf{x} - \boldsymbol{\tau}(t))$. The optimal solution F_θ^* corresponds to the conditional expectation over the posterior trajectory distribution given the current state $\mathbf{x}_t$:

$$F_\theta^*(\mathbf{x}_t, t \mid \mathbf{x}_s, \mathbf{x}_g) = \mathbb{E}_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau} \mid \mathbf{x}_t, t, \mathbf{x}_s, \mathbf{x}_g)} \left[\dot{\boldsymbol{\tau}}(t) - k(\mathbf{x}_t - \boldsymbol{\tau}(t))\right]. \tag{12}$$

Bayesian Interpretation. Crucially, the posterior distribution $p(\boldsymbol{\tau} \mid \mathbf{x}_t, t, \mathbf{x}_s, \mathbf{x}_g)$ is derived via Bayes' rule:

$$p(\boldsymbol{\tau} \mid \mathbf{x}_t, t, \mathbf{x}_s, \mathbf{x}_g) \propto p(\mathbf{x}_t \mid \boldsymbol{\tau}, t) \cdot p_{\text{data}}(\boldsymbol{\tau} \mid \mathbf{x}_s, \mathbf{x}_g). \tag{13}$$

Since the support of $p_{\text{data}}(\boldsymbol{\tau} \mid \mathbf{x}_s, \mathbf{x}_g)$ lies on $\mathcal{T}_{\mathbf{x}_s,\mathbf{x}_g}$, this implies that the support of the posterior distribution $p(\boldsymbol{\tau} \mid \mathbf{x}_t, t, \mathbf{x}_s, \mathbf{x}_g)$ also lies on $\mathcal{T}_{\mathbf{x}_s,\mathbf{x}_g}$, meaning the model considers *only* those trajectories that connect the current state $\mathbf{x}_t$ to the goal $\mathbf{x}_g$.

The flow matching theorem (Theorem 1 of Lipman et al. (2023)), as applied to streaming flow matching (Jiang et al., 2025) states that the optimal vector field F_θ^* is the expectation of the conditional vector field $F_{\boldsymbol{\tau}}$ dynamics over the posterior trajectory distribution:

$$F_\theta^*(\mathbf{x}_t, t \mid \mathbf{x}_s, \mathbf{x}_g) = \int_{\mathcal{T}_{\mathbf{x}_s,\mathbf{x}_g}} F_{\boldsymbol{\tau}}(\mathbf{x}_t, t)\, p(\boldsymbol{\tau} \mid \mathbf{x}, t, \mathbf{x}_s, \mathbf{x}_g)\, d\boldsymbol{\tau} \tag{14}$$

$$= \int_{\mathcal{T}_{\mathbf{x}_s,\mathbf{x}_g}} (\dot{\boldsymbol{\tau}}(t) - k(\mathbf{x}_t - \boldsymbol{\tau}(t)))\, p(\boldsymbol{\tau} \mid \mathbf{x}, t, \mathbf{x}_s, \mathbf{x}_g)\, d\boldsymbol{\tau} \tag{15}$$

Furthermore, the probability density of states p_θ^* sampled from the optimal flow F_θ^* is a weighted average of the probability densities of the conditional flows, averaged over the posterior trajectory distribution. From Theorem 2 of (Jiang et al., 2025):

$$p_\theta^*(\mathbf{x}_t \mid \mathbf{x}_t, t, \mathbf{x}_s, \mathbf{x}_g) = \int_{\mathcal{T}_{\mathbf{x}_s,\mathbf{x}_g}} p(\mathbf{x}_t \mid \boldsymbol{\tau}, t)\, p(\boldsymbol{\tau} \mid \mathbf{x}_t, t, \mathbf{x}_s, \mathbf{x}_g)\, d\boldsymbol{\tau} \tag{16}$$

$$= \int_{\mathcal{T}_{\mathbf{x}_s,\mathbf{x}_g}} \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\tau}(t), \sigma_t^2 I\right)\ p(\boldsymbol{\tau} \mid \mathbf{x}_t, t, \mathbf{x}_s, \mathbf{x}_g)\, d\boldsymbol{\tau} \quad \text{where } \sigma_t = \sigma_0 e^{-kt} \tag{17}$$

Equation (17) implies that the planner essentially performs a weighted average of Gaussian convolutions of all feasible trajectories $\boldsymbol{\tau}$ that pass through $\mathbf{x}_t$ at time t, weighted by the posterior probability of the trajectories.

Terminal Convergence. As $t \to T$, the Gaussian convolution at the terminal time T is performed by standard deviation $\sigma_T = \sigma_0 e^{-kT}$. For reasonably large values of k and T, we have that $\sigma_T \approx 0$. Therefore, at $t = T$, the sampled trajectories form an extremely narrow Gaussian distribution centered at $\mathbf{x}_g$. Consequently, the posterior $p(\boldsymbol{\tau} \mid \mathbf{x}_T, T, \mathbf{x}_s, \mathbf{x}_g)$ also forms a peaked distribution at $\mathbf{x}_g$.

Assuming that demonstrations terminate in the goal state ($\boldsymbol{\tau}(T) = \mathbf{x}_g$) at rest ($\dot{\boldsymbol{\tau}}(T) \approx 0$), the optimal vector field from (15) becomes a global linear sink:

$$F_\theta^*(\mathbf{x}, T \mid \mathbf{x}_s, \mathbf{x}_g) = \int_{\mathcal{T}_{\mathbf{x}_s, \mathbf{x}_g}} (\dot{\boldsymbol{\tau}}(T) - k(\mathbf{x} - \boldsymbol{\tau}(T)))\; p(\boldsymbol{\tau} \mid \mathbf{x}, T, \mathbf{x}_s, \mathbf{x}_g)\, d\boldsymbol{\tau} \tag{18}$$

$$\approx -k(\mathbf{x} - \mathbf{x}_g) \tag{19}$$

This shows that under ideal optimization of the streaming flow matching objective, the vector field at the final time step forms a contracting basin of attraction around the goal $\mathbf{x}_g$, providing theoretical support for goal-reaching behavior when the optimization error remains bounded.

D Implementation Details

Both F_θ^p and F_θ^v are conditional 1-D U-Nets with sinusoidal time embedding. For the DT-HOCBF, we use linear class-$\mathcal{K}$ functions $\alpha_{i,1}(s) = k_{i,1}s$, $\alpha_{i,2}(s) = k_{i,2}s$ with $k_{i,1} = 0.08$, $k_{i,2} = 0.15$, safety margin $\delta = 0.01$, and solve the QP via SLSQP.

E Computational Analysis

Table 3: Computational efficiency on Maze2D Large.

Method	NN (ms)	t_{QP} (ms)	t_{total} (ms)	MFLOPs/step	#QP
Diffuser	3.6	–	1052.3	771.1	0
SafeDiffuser	3.8	35.6	10226.5	771.1	$H \times K$
FM	3.5	–	1009.1	771.1	0
SafeFM	3.9	35.7	10245.0	771.1	$H \times K$
FlowMatcher	3.5	–	1014.8	771.1	0
SafeFlowMatcher	3.8	35.8	10267.0	771.1	$H \times K$
StreamingFlow (open-loop)	4.8	–	1864.8	120.9	0
StreamingFlow (closed-loop)	5.6	–	2163.2	120.9	0
`SafeStreamingFlow` (open-loop)	4.8	4.4	3536.2	120.9	H
`SafeStreamingFlow` (closed-loop)	5.5	4.8	4001.0	120.9	H

Although StreamingFlow has higher wall-clock time than trajectory-level baselines in the nominal setting, this is because sequential generation cannot exploit batched parallelism. However, the per-cycle FLOPs are $6.4\times$ lower (120.9 vs. 771.1 MFLOPs/step) due to the compact factorized architecture. For safe variants, QP overhead dominates: trajectory-level methods require $\mathcal{O}(H \times K)$ solves while `SafeStreamingFlow` requires only $\mathcal{O}(H)$, yielding $\sim 3\times$ lower end-to-end latency. Closed-loop operation is slower than open-loop because the current state is updated from the environment observation at every execution cycle, introducing additional in-place memory operations for state reassignment and conditioning refresh that are absent in open-loop rollouts. Nevertheless, trajectory-level planners must regenerate the full H-horizon trajectory at every closed-loop replanning cycle, compounding their cost; `SafeStreamingFlow` incurs only a modest per-cycle overhead.