NOISE-SPACE SAFETY CORRECTOR FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models have demonstrated strong generalization capabilities across diverse robotic manipulation tasks. However, physical deployment of these models requires satisfying safety constraints while retaining task performance. Existing approaches enforce safety by correcting actions during sampling or projecting them after generation. Although these approaches produce safe actions, they either ignore chunk level dependencies within action sequences or push actions away from the learned behavior of the pretrained policy, thereby degrading task performance. To address these limitations, we propose Noise Safe Corrector (NSC), a test-time safety filter that corrects the noise of a pretrained VLA policy. As the transformation from the noise to the action chunk is deterministic, we can optimize the noise to enforce safety constraints while taking chunk level dependencies into account and preserving pretrained knowledge encoded in the policy. To enforce safety constraints, we use a control barrier function (CBF) formulated in noise space and iteratively solve a quadratic program (QP), which has a closed-form solution. We further prove that NSC obtains a safe action chunk within a finite number of iterations under mild assumptions. Experiments across three simulation environments and real-world robotic tasks show that NSC outperforms baselines in terms of task success rate while satisfying safety constraints.

1 INTRODUCTION

Vision-Language-Action (VLA) models have demonstrated remarkable capabilities in instruction following, dexterous manipulation, and long-horizon task execution (Brohan et al., 2023,Kim et al., 2024, Intelligence et al., 2025, Bjorck et al., 2025). However, deploying these models in the real world requires careful consideration of safety. For example, humans or obstacles may unexpectedly appear in the path of the robot, and a collision can lead to severe injury or hardware damage. Therefore, a deployed VLA model must not only follow the instruction but also satisfy safety constraints. While a naive way to enforce safety is to directly fine-tune VLA models with safety objectives (Zhang et al., 2025), it requires expensive retraining and typically handles safety as a soft constraint, which prevents formal safety guarantees.

A promising alternative is to use test-time safety filters, which modify the actions generated by the pretrained policy during execution without retraining. In particular, we can incorporate control barrier function (CBF) to guarantee safety by projecting the original action onto the set of actions satisfying the CBF condition through a quadratic program (QP). Prior works apply these filters after action generation (Hu et al., 2026) or to intermediate actions of the generative process (Xiao et al., 2025; Yang et al., 2026), improving safety while maintaining instruction-following capabilities.

Despite their success, we find that current approaches often fail to complete the task in environments that differ substantially from training. As these filters constrain each action independently and do not coordinate corrections across the action chunk, they fail to find a detour around obstacles that block the path. In addition, directly modifying actions pushes them away from the learned behavior of the pretrained policy, which degrades task performance. Therefore, safety filters for VLA policies should reason over the entire action chunk while staying within the learned behavior of the policy.

To address these challenges, we introduce Noise Safe Corrector (NSC), a test-time safety filter that corrects the entire action chunk simultaneously to guarantee safety. Our key idea is to correct the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

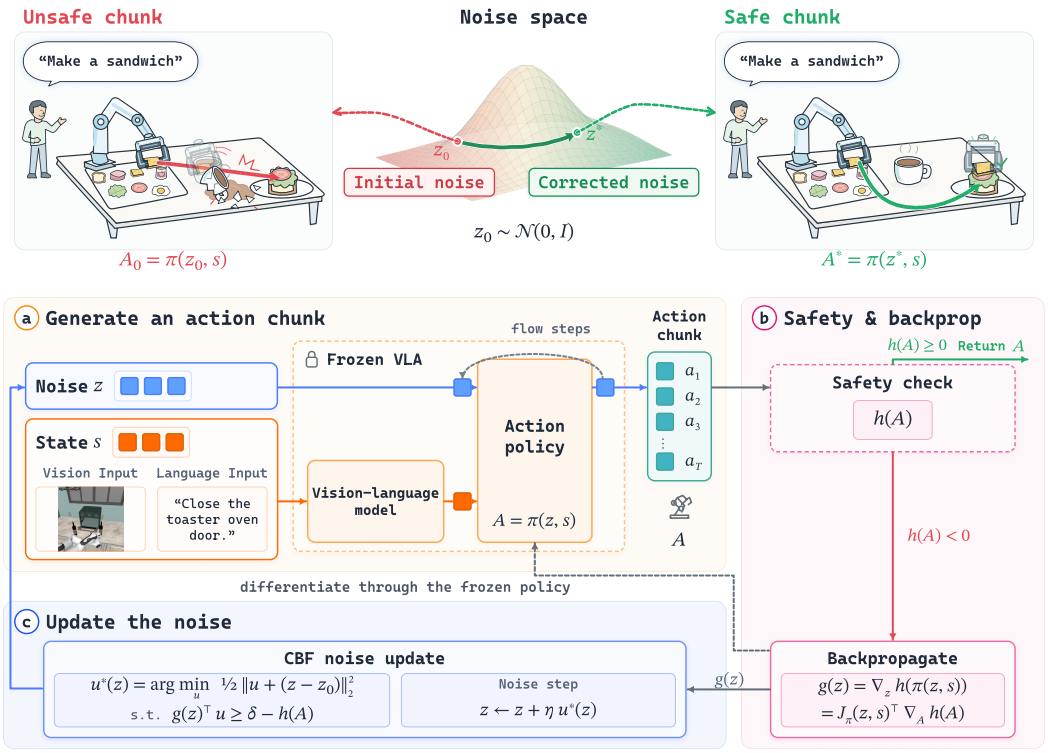


Figure 1: Overview of Noise Space Corrector (NSC). Our method iterates three stages to correct the action chunk in the noise space. First, it generates action chunk given a noise. Then, we evaluate safety and backpropagate gradient through the flow policy. Finally, we solve QP to update the noise.

initial noise of the pretrained policy rather than the actions, as illustrated in Figure 1. Since a flow-based policy deterministically maps the noise to the action chunk, we can evaluate the safety of the whole chunk and correct it by modifying the noise. Specifically, NSC first **(a) generates** the action chunk from the initial noise, **(b) evaluates** safety and compute gradient, **(c) updates** the noise by solving a QP with a closed-form solution, and repeat the process. Since the final actions are always regenerated by the frozen policy, NSC preserves the learned behavior of the pretrained policy.

We first prove that our approach provides a finite-time guarantee on safety constraints under smoothness and gradient non-degeneracy assumptions. Then, we evaluate NSC on two benchmarks, Safe-LIBERO (Hu et al., 2026) and VLA-Arena (Zhang et al., 2026), confirming that our method achieves better task success rate and safety violation rate compared to the baselines. We further propose evaluate NSC on SafeDetour, a new evaluation environment based on RoboCasa (Nasiriany et al., 2024) that explicitly requires the robot to find a detour to complete the task. Finally, we conduct real-robot experiments in environments with obstacles and demonstrate that NSC outperforms the baselines in safe task completion.

2 Related Works

Vision Language Action Models. Vision-Language-Action (VLA) models have shown strong performance across diverse robotic manipulation tasks (Brohan et al., 2023; Intelligence et al., 2025; Kim et al., 2024). Training on large-scale heterogeneous robot datasets together with vision-language data has improved generalization across tasks, objects, environments, and robot embodiments (O'Neill et al., 2024; Reed et al., 2022). Recent VLA models further demonstrate capabilities in instruction following, dexterous manipulation, and long-horizon task execution (Intelligence et al., 2025; Bjorck et al., 2025; Xiaomi Robotics Team et al., 2026). For real-world deployment, however, robot policies must satisfy both safety and task completion for real-world deployment. Without consideration of safety, the policies are vulnerable to failures such as collisions with unseen

obstacles Safety requires the system to remain within a prescribed safe set during execution (Ames et al., 2019; Alpern & Schneider, 1985), but current VLA do not explicitly enforce these properties.

Safety Control. Safety control for VLA models aims to satisfy constraints while preserving task performance. Safe reinforcement learning (SafeRL) commonly optimizes task reward subject to bounds on expected cumulative safety costs (Achiam et al., 2017; Zhang et al., 2025). However, these approaches require additional policy training and data collection, and do not generally guarantee constraint satisfaction at every action step. Test-time safety filters impose constraints on pretrained VLA without additional policy training. Control barrier function (Ames et al., 2019) provides a principled basis for safety filters by expressing safety requirements as constraints. Recent approaches apply CBF constraints to pretrained diffusion, flow-matching, and VLA policies, correcting actions or trajectories during or after sampling (Xiao et al., 2025; Yang et al., 2026; Hu et al., 2026). These methods enforce safety by projecting actions or sampling updates onto the feasible set defined by safety constraints. However, these methods can lead to deadlock by enforcing only single action-level safety or degrade task performance by directly projection the action from the policy.

3 PRELIMINARY

3.1 FLOW MATCHING ACTION POLICIES

Let $s \in \mathcal{S}$ denote the policy input, including visual observations, language instructions, and robot states, where $\mathcal{S}$ is the policy input space. An action chunk consists of T actions $a_t \in \mathbb{R}^n$, where n is the dimension of each action and $t = 1, \dots, T$. We represent the chunk as the stacked vector $A = (a_1^\top, \dots, a_T^\top)^\top \in \mathbb{R}^{Tn}$.

A flow matching policy generates an action chunk from a noise input $z \in \mathbb{R}^m$, where m is the noise dimension. It uses a conditional velocity field $v_\theta(\cdot, \xi; s) : \mathbb{R}^m \to \mathbb{R}^m$, with parameters θ learned through a flow matching objective. Here $\xi \in [0, 1]$ denotes the sampling time of the flow.

At inference, the policy samples $z \sim \mathcal{N}(0, I_m)$, and evolves an intermediate sample $X_\xi \in \mathbb{R}^m$ according to

$$\frac{dX_\xi}{d\xi} = v_\theta(X_\xi, \xi; s), \qquad X_0 = z. \tag{1}$$

We denote the complete mapping from the initial noise to the action chunk, including any output transformation applied to the final flow sample X_1, by $\pi : \mathbb{R}^m \times \mathcal{S} \to \mathbb{R}^{Tn}$. Thus, the generated action chunk is

$$A = \pi(z, s). \tag{2}$$

Appendix D provides further details on flow matching policies.

3.2 CONTROL BARRIER FUNCTIONS

Consider the control-affine system and safe set

$$\dot{x} = f(x) + G(x)u, \qquad \mathcal{C} = \{x \in \mathcal{D} \mid h(x) \geq 0\},$$

where $\mathcal{D} \subseteq \mathbb{R}^{d_x}$ is open, $x \in \mathcal{D}$ is the system state, $u \in \mathcal{U} \subseteq \mathbb{R}^{d_u}$ is the control input, f and G are locally Lipschitz, and $h \in C^1(\mathcal{D})$ defines the safe set $\mathcal{C}$.

The function h is a control barrier function (CBF) if there exists a locally Lipschitz extended class-$\mathcal{K}$ function α such that

$$\sup_{u \in \mathcal{U}} \nabla h(x)^\top \big(f(x) + G(x)u\big) \geq -\alpha\big(h(x)\big), \qquad \forall x \in \mathcal{D}, \tag{3}$$

where α is strictly increasing and satisfies $\alpha(0) = 0$.

Under the condition $\nabla h(x) \neq 0$ on $\partial\mathcal{C}$, any locally Lipschitz feedback satisfying

$$\nabla h(x)^\top \big(f(x) + G(x)u\big) + \alpha\big(h(x)\big) \geq 0 \tag{4}$$

renders $\mathcal{C}$ forward invariant (Ames et al., 2019; 2016). Thus, a trajectory initialized in $\mathcal{C}$ remains in $\mathcal{C}$ for as long as the solution exists. If the closed-loop system is forward complete, the safety guarantee holds for all future time.

CBF Quadratic Programs. A quadratic program (QP) minimizes a quadratic objective subject to affine constraints. At a fixed state x, the CBF constraint is affine in u. For $\mathcal{U} = \mathbb{R}^{d_u}$, the CBF safety filter solves the following QP to find the control closest to a reference input $u_{\text{ref}}(x)$ while satisfying the CBF constraint:

$$\begin{aligned} u^\star(x) = \underset{u \in \mathbb{R}^{d_u}}{\arg\min} \quad & \frac{1}{2} \|u - u_{\text{ref}}(x)\|_2^2 \\ \text{s.t.} \quad & \nabla h(x)^\top \big(f(x) + G(x)u\big) + \alpha\big(h(x)\big) \geq 0. \end{aligned} \tag{5}$$

With $G(x)^\top \nabla h(x) \neq 0$, the QP has the unique closed-form solution (Ames et al., 2016)

$$u^\star(x) = u_{\text{ref}}(x) + \frac{[-\psi(x)]_+}{\|G(x)^\top \nabla h(x)\|_2^2} G(x)^\top \nabla h(x), \tag{6}$$

where $\psi(x) = \nabla h(x)^\top \big(f(x) + G(x)u_{\text{ref}}(x)\big) + \alpha\big(h(x)\big)$, and $[v]_+ = \max\{v, 0\}$. If $\psi(x) \geq 0$, then $u^\star(x) = u_{\text{ref}}(x)$. Otherwise, $u^\star(x)$ is the input closest to $u_{\text{ref}}(x)$ that satisfies the CBF constraint. A detailed derivation of the closed-form solution is provided in Appendix C.

4 METHOD

4.1 PROBLEM FORMULATION

We consider robotic manipulation with a pretrained VLA conditioned on visual observations, robot state, and a language instruction. Let $s \in \mathcal{S}$ denote this conditioning. At each policy query, the model predicts T future actions $a_t \in \mathbb{R}^n$, where n is the action dimension. We represent the action chunk by stacking these actions as $A = (a_1^\top, \ldots, a_T^\top)^\top \in \mathbb{R}^{Tn}$. Safety is specified by a continuously differentiable function $h : \mathbb{R}^{Tn} \to \mathbb{R}$, where $h(A) \geq 0$ indicates that the entire predicted chunk satisfies the safety constraints.

Objective. Let $z \in \mathbb{R}^m$ denote the noise input, where m is the noise dimension. We seek a safe action chunk by adjusting the noise input of the frozen policy $\pi : \mathbb{R}^m \times \mathcal{S} \to \mathbb{R}^{Tn}$. The policy generates $A = \pi(z, s)$, and we assume that $\pi(\cdot, s)$ is deterministic and continuously differentiable. We formulate noise correction as the following constrained optimization problem:

$$\begin{aligned} \min_{z \in \mathbb{R}^m} \quad & \frac{1}{2} \|z - z_0\|_2^2 \\ \text{subject to} \quad & h\big(\pi(z, s)\big) \geq 0, \end{aligned} \tag{7}$$

where $z_0 \sim \mathcal{N}(0, I_m)$ is the initial noise sample and $A_0 = \pi(z_0, s)$ is the initial proposal. The objective penalizes deviations from z_0, while the constraint requires the generated chunk to satisfy the safety constraint.

4.2 NOISE SPACE SAFETY CORRECTION

NSC imposes a CBF constraint on noise updates to increase the safety score of the generated action chunk. At each correction step, NSC solves a quadratic program to obtain an update direction, updates the noise, and regenerates the action chunk through the frozen policy. With s and h fixed, we define the noise-space safety function $H(z)$ and its gradient $g(z)$ as

$$H(z) = h\big(\pi(z, s)\big), \qquad g(z) = \nabla_z H(z) = J_\pi(z, s)^\top \nabla_A h\big(\pi(z, s)\big), \tag{8}$$

where $J_\pi(z, s) \in \mathbb{R}^{Tn \times m}$ is the policy Jacobian with respect to the noise. Backpropagating the chunk safety gradient through the policy computes $g(z)$ with one vector–Jacobian product.

Let $u \in \mathbb{R}^m$ denote an update direction for z. For an unsafe candidate, we impose the CBF constraint $g(z)^\top u \geq \delta - H(z)$, where $\delta > 0$ sets a positive margin on $g(z)^\top u$ while $H(z) < 0$.

At the current noise z, the negative gradient of the objective in Equation (7), $-(z - z_0)$, serves as the reference direction.

Algorithm 1 Noise Safe Corrector (NSC)

Require: Policy π, conditioning s, safety function h, step size η, margin δ, update budget K
1: Sample $z_0 \sim \mathcal{N}(0, I_m)$ and set $z \leftarrow z_0$
2: **for** $k = 0, \ldots, K$ **do**
3: $A \leftarrow \pi(z, s)$
4: **if** $h(A) \geq 0$ **then return** A
5: $g(z) \leftarrow \nabla_z h(\pi(z, s))$
6: Compute $u^\star(z)$ using Equation (10)
7: $z \leftarrow z + \eta u^\star(z)$
8: **end for**

We select the direction u closest to this reference while satisfying the CBF constraint by solving

$$\begin{aligned} \min_{u \in \mathbb{R}^m} \quad & \frac{1}{2}\|u + (z - z_0)\|_2^2 \\ \text{subject to} \quad & g(z)^\top u \geq \delta - H(z). \end{aligned} \tag{9}$$

For $H(z) < 0$, the QP is feasible if and only if $g(z) \neq 0$. For $g(z) \neq 0$, the unique solution $u^\star(z)$ decomposes into a component along $g(z)$ that satisfies the CBF constraint and an orthogonal component that pulls the noise toward z_0.

$$u^\star(z) = \frac{g(z)^\top u^\star(z)}{\|g(z)\|_2^2}\, g(z) - P_{g^\perp}(z - z_0), \tag{10}$$

where $g(z)^\top u^\star(z) = \max\{\delta - H(z), -g(z)^\top (z - z_0)\}$ is the directional derivative of H along the update direction, and $P_{g^\perp} = I_m - g(z)g(z)^\top / \|g(z)\|_2^2$ is the orthogonal projector onto the subspace perpendicular to $g(z)$. A detailed derivation of the closed-form solution to the CBF-QP is provided in Appendix C.1.

NSC uses $u^\star(z)$ to update the noise as $z \leftarrow z + \eta u^\star(z)$ with step size $\eta > 0$, then regenerates $A = \pi(z, s)$. If the updated chunk remains unsafe, NSC recomputes $H(z)$ and $g(z)$ and repeats the correction until $H(z) \geq 0$ or the number of noise updates reaches the fixed update budget K. Appendix A.2 provides the assumptions and proof for finite-step safety recovery. Algorithm 1 provides the complete procedure.

Safety guarantees. To analyze safety recovery, we consider moving z continuously along $u^\star(z)$ according to $dz/d\tau = u^\star(z)$ with $z(0) = z_0$. Here τ denotes optimization time, distinct from robot execution time and the policy's internal sampling time. By the chain rule, the derivative of the safety score is $dH/d\tau = g(z)^\top u^\star(z)$. The CBF constraint therefore implies $dH/d\tau \geq \delta - H(z)$, yielding the Proposition 4.1.

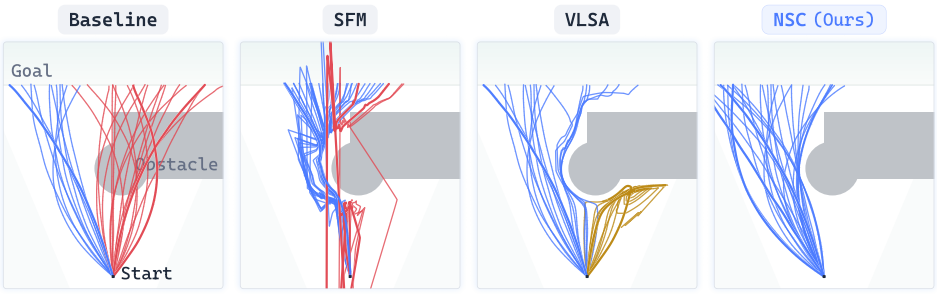
Proposition 4.1 (Finite-time safety recovery)**.** *Let $u^\star$ be given by Equation (10) with $\delta > 0$. Suppose $H(z_0) < 0$ and ∇H is locally Lipschitz and uniformly bounded away from zero on the recovery neighborhood defined in Appendix A.1. Then the solution $z(\tau)$ of $dz/d\tau = u^\star(z)$ with $z(0) = z_0$ reaches $H \geq 0$ in finite optimization time. For $0 \leq \tau \leq \tau_{\text{hit}}$, Until the first safe point,*

$$\begin{aligned} H(z(\tau)) &\geq \delta + \big(H(z_0) - \delta\big)e^{-\tau}, \\ \tau_{\text{hit}} &\leq \log\left(1 - \frac{H(z_0)}{\delta}\right), \end{aligned} \tag{11}$$

where τ_{hit} is the first optimization time at which $H(z(\tau)) \geq 0$. See proof in Appendix A.1.

5 Experiments

In these section, we validate the superiority of NSC through extensive experiments. First, we conduct an experiment on a controlled 2D toy environment to clearly demonstrate our motivation (§5.2). Then, we conduct experiments on simulated robotic manipulation tasks, including SafeDetour, a new environment that tests whether the robot choose a detour for safe success (§5.3 and §5.4). Finally, we evaluate NSC on real-world robotic manipulation (§5.5). Through these experiments, we assess whether noise-space correction improves task performance while maintaining safety. Further experimental details are provided in Appendix B.



	Baseline	SFM	VLSA	NSC (Ours)
TSR	100%	100%	58%	100%
CAR	31%	79%	100%	100%
SS	31%	79%	58%	100%

Figure 2: **Comparison of test-time safety methods with the right route blocked.** We show representative trajectories and report task success rate (TSR), collision avoidance rate (CAR), and safe success rate (SS) over 200 trials per method. Blue, red, and yellow trajectories indicate collision-free success, success with collisions, and collision-free failure, respectively.

5.1 Experimental Setup

Baselines. Existing training-free CBF filters for VLAs correct either the action or the generation process. We therefore compare NSC with (i) **VLSA** (Hu et al., 2026), which corrects the current action with a CBF-QP, and (ii) **SafeFlowMatcher (SFM)** (Yang et al., 2026), which perturbs the flow's vector field with a CBF-QP. All methods use the same barrier function h in each environment.

Metrics. Following Hu et al. (2026), the task success rate (TSR) is the percentage of episodes that complete the task within the horizon, and the collision avoidance rate (CAR) is the percentage of collision-free episodes. The safe success rate (SS) is the percentage of episodes that are both collision-free and successful.

5.2 Toy Setting

To build intuition for safety correction, we conduct a toy study on a 2D navigation task. We train a flow-matching policy to navigate from a start point to a goal region in a 2D environment without obstacles. At test time, we introduce an obstacle that blocks the right route and compare the resulting trajectories of NSC with those of the base policy, SFM, and VLSA. See Appendix B.2 for details.

As shown in Figure 2, NSC reaches safe success in 100% of rollouts, whereas the best performing baseline do so in 79%. Notably, the baselines exhibit two failure modes. (1) VLSA cannot reach the goal when the base policy initially selects the unsafe right route. It avoids collisions by correcting the actions after generation, but this pushes them away from the learned behavior of the pretrained policy. (2) SFM collides with an obstacle. It corrects each action independently without coordinating the corrections across the action chunk, producing whole action chunks that violate safety constraints (red trajectories). In contrast to these failures, NSC evaluates the whole chunk and regenerates it through the frozen pretrained policy, so it commits to the choose the left route before reaching the obstacle. Overall, these results show that correcting the noise at the chunk level lets the policy find a safe route while staying within its learned behavior.

5.3 Simulation Benchmarks

Task environments. We evaluate NSC and baselines in two simulated VLA safety benchmarks. **SafeLIBERO** (Hu et al., 2026) adds obstacles to 16 LIBERO tasks at two levels, near the target object (Level I) or directly on the movement path (Level II). We use the more challenging Level II, with 50 episodes per task. **VLA-Arena** (Zhang et al., 2026) grades its tasks into three difficulty levels (L0–L2). We use StaticObstacles, DynamicObstacles, and HazardAvoidance under the standard protocol, which fine-tunes the policy on L0 and evaluates on L0–L2 with 10 episodes per task.

Table 1: **Results on the SafeLIBERO benchmark.** Results are reported as mean ± standard deviation over three runs. Higher is better. The best and second-best results are **bold** and underlined.

	Spatial			Object			Goal			Long			Average		
Method	TSR	CAR	**SS**	TSR	CAR	**SS**	TSR	CAR	**SS**	TSR	CAR	**SS**	TSR	CAR	**SS**
$\pi_{0.5}$	58±2	14±1	13±1	73±3	23±1	21±1	68±4	34±1	31±2	51±2	16±1	14±1	63±2	21±1	20±1
+ SFM	79±0	83±1	74±1	82±1	81±2	70±1	66±1	53±1	44±2	50±1	75±3	38±1	69±1	73±0	57±1
+ VLSA	77±2	75±1	70±1	73±2	**92±1**	68±2	80±0	56±2	49±2	34±3	**86±2**	31±2	66±0	77±1	55±1
+ NSC	**89±2**	**87±1**	**84±1**	**86±2**	88±3	**77±1**	**82±1**	**61±1**	**56±2**	**54±4**	75±5	**47±1**	**78±1**	**78±2**	**66±0**
GR00T N1.7	42±2	12±1	12±1	49±2	43±2	15±1	**28±1**	1±1	0±1	**21±1**	6±1	3±1	35±0	15±1	8±0
+ SFM	67±1	**87±1**	62±1	37±2	**94±1**	34±3	14±1	58±1	5±0	9±1	**84±4**	9±1	32±1	**81±1**	27±1
+ VLSA	56±1	72±1	51±1	44±1	92±1	39±2	27±2	53±2	**20±2**	3±1	81±1	2±1	33±1	74±1	28±1
+ NSC	**72±1**	77±2	**63±1**	**54±2**	90±0	**48±2**	18±1	**73±1**	16±1	18±2	82±2	**17±1**	**40±0**	80±0	**36±1**

Table 2: **Results on the VLA-Arena benchmark.** Results are reported as mean ± standard deviation over three runs. Higher is better. The best and second-best results are **bold** and underlined.

	Static Obstacle			Dynamic Obstacle			Hazard Avoidance			Average		
Method	TSR	CAR	SS	TSR	CAR	SS	TSR	CAR	SS	TSR	CAR	SS
$\pi_{0.5}$	**59 ± 3**	50 ± 1	34 ± 1	37 ± 2	47 ± 5	24 ± 1	36 ± 3	18 ± 4	5 ± 1	44 ± 2	38 ± 3	21 ± 1
+ SFM	53 ± 3	73 ± 3	53 ± 3	26 ± 3	66 ± 4	24 ± 4	28 ± 2	**35 ± 1**	4 ± 1	36 ± 2	58 ± 1	27 ± 2
+ VLSA	36 ± 5	**91 ± 1**	33 ± 5	34 ± 2	61 ± 1	32 ± 1	20 ± 2	20 ± 3	4 ± 1	30 ± 3	57 ± 1	23 ± 2
+ NSC	57 ± 1	83 ± 2	**56 ± 1**	**43 ± 1**	**73 ± 1**	**37 ± 1**	**40 ± 1**	23 ± 1	**8 ± 1**	**46 ± 1**	**60 ± 1**	**34 ± 1**

Base policies. All filters wrap frozen flow-matching VLAs. On SafeLIBERO, we use $\pi_{0.5}$ (Intelligence et al., 2025) and GR00T N1.7 (Bjorck et al., 2025), both fine-tuned on LIBERO, as base policies. On VLA-Arena, we use $\pi_{0.5}$ fine-tuned on VLA-Arena-L0 tasks (Zhang et al., 2026) as the base policy. Each VLA model used in our experiments uses a flow-matching action policy that deterministically maps an initial Gaussian noise sample to an action.

Results. Tables 1 and 2 summarize the results on SafeLIBERO and VLA-Arena. NSC achieves the highest average TSR and SS on both benchmarks. It also attains the highest average CAR in all settings except GR00T N1.7 on SafeLIBERO, where it ranks second behind SafeFlowMatcher (SFM). In particular, NSC outperforms the baseline filters by a large margin in SS on SafeLIBERO Long and the HazardAvoidance suite of VLA-Arena, where accounting for unsafety beyond the immediate action is important for safe task completion. Overall, these results indicate that correcting entire action chunks in the noise space of the VLA policy improves safety without degrading the instruction-following ability of the pretrained model.

We also conduct experiments with GR00T N1.7 as the base policy on SafeLIBERO to validate the robustness of NSC with respect to the choice of base policy. As shown in Table 1, NSC consistently outperforms the baselines with a different backbone.

5.4 SafeDetour

Task environments. SafeDetour tests whether a filter can anticipate hazards later in the action chunk. Each scene offers two routes that complete the task (Figure 3). (i) a *trap route*, which the policy prefers and which is collision-free at first but leads into a hazard, so a filter that only avoids immediate collisions ends in a collision or a deadlock; and (ii) a *detour route*, which is collision-free throughout. We instantiate SafeDetour in three RoboCasa tasks. In each, we place obstacles so that avoiding only the nearest one along the trap route leads into another. **Cabinet**: the robot places an object in a cabinet with glasses at the front and a wine glass at the policy's usual placement spot. **Stove**: the robot places an object in a pan on a rear burner, with a kettle and a pot staggered on the front burners. **Toaster**: the robot closes an open toaster-oven door, which the policy usually lifts at

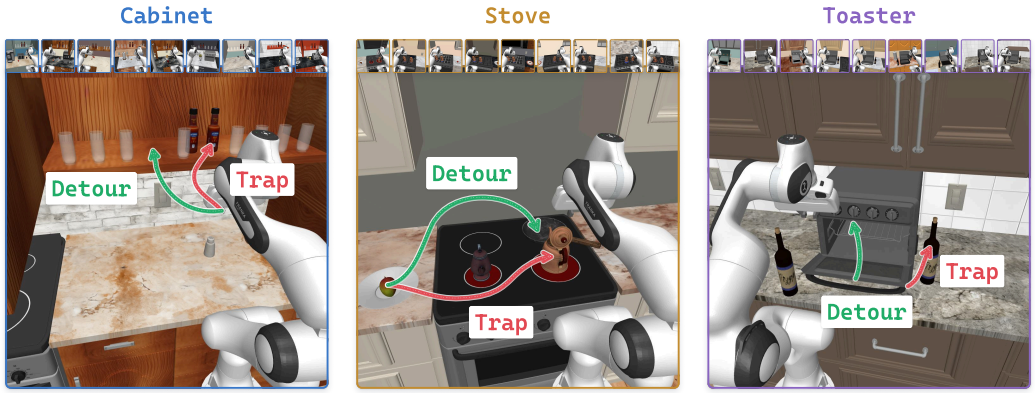


Figure 3: **SafeDetour tasks.** Red: trap route, preferred by the policy but leading into a hazard later. Green: detour route, collision-free.

Table 3: **Results on the SafeDetour.** Results are reported as mean ± standard deviation across the 30 scenes of each task. Higher is better. The best and second-best results are **bold** and underlined.

	Cabinet			Stove			Toaster			Average		
Method	TSR	CAR	SS	TSR	CAR	SS	TSR	CAR	SS	TSR	CAR	SS
Base	**97 ± 9**	8 ± 16	7 ± 16	**81 ± 23**	13 ± 21	10 ± 18	**93 ± 14**	15 ± 14	14 ± 14	**90 ± 18**	12 ± 17	10 ± 16
+ SFM	75 ± 25	62 ± 29	46 ± 29	35 ± 28	**66 ± 33**	27 ± 27	55 ± 32	92 ± 14	51 ± 31	55 ± 32	73 ± 29	41 ± 31
+ VLSA	95 ± 14	21 ± 24	18 ± 20	51 ± 33	49 ± 37	14 ± 20	91 ± 15	49 ± 29	43 ± 28	79 ± 30	40 ± 33	25 ± 26
+ **NSC**	95 ± 10	**71 ± 25**	**67 ± 26**	**81 ± 22**	55 ± 21	**47 ± 24**	91 ± 14	**96 ± 11**	**87 ± 17**	89 ± 17	**74 ± 26**	**67 ± 28**

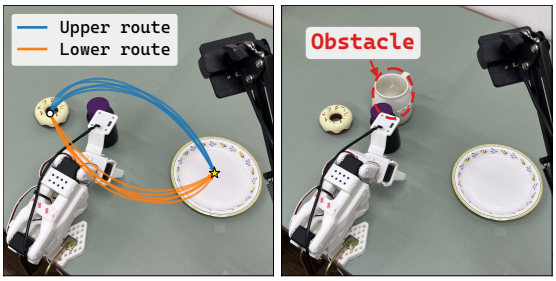
a corner; a bottle stands at one corner, and another stands behind the path to the opposite corner. Appendix B.4 details the scene layouts and both routes for each task.

Base policies. On SafeDetour, we use the 5B variant of Xiaomi-Robotics-1 (Xiaomi Robotics Team et al., 2026), a VLA model pretrained on over 100,000 hours of real-world manipulation trajectories spanning more than 1,700 scenarios. This large and diverse pre-training corpus, followed by post-training on cross-embodiment robot data, supports generalization to unseen environments and strong downstream performance on RoboCasa (Nasiriany et al., 2026; 2024). We choose this strong base policy to reduce the influence of policy limitations when evaluating the filters. We use the same baselines and metrics as in Section 5.3.

Results. Table 3 summarizes the results on SafeDetour. NSC achieves the highest safe success across all three tasks. Although NSC and SFM attain similar average collision avoidance rates, NSC achieves a substantially higher task success rate. These results are consistent with two aspects of NSC, looking ahead later hazards and using the behavior of the pretrained model to complete the task safely. By evaluating safety over the entire predicted chunk, NSC accounts for obstacles further along the trap route and can redirect earlier actions before the robot reaches them. By correcting the initial noise and regenerating the full chunk through the frozen policy, NSC can draw on the policy's learned behaviors to generate an alternative approach or placement. This allows the correction to address later hazards while retaining the coordinated actions needed for task completion.

5.5 Real-World Robots

We evaluate the NSC on real-world robot manipulation using SO101 arm (Knight et al., 2024). We design the **PickandPlace** task with two available routes: an upper route and a lower route. We



(a) Training setup (b) Evaluation setup

Figure 4: **Setup for the real-world experiments.**

Table 4: **Quantitative results in real-world experiments.** Results are reported with 10 episodes.

Method	TSR ↑	CAR ↑	SS ↑
$\pi_{0.5}$ (Base)	10/10	0/10	0/10
SFM	10/10	0/10	0/10
VLSA	0/10	10/10	0/10
NSC (Ours)	**10/10**	**10/10**	**10/10**

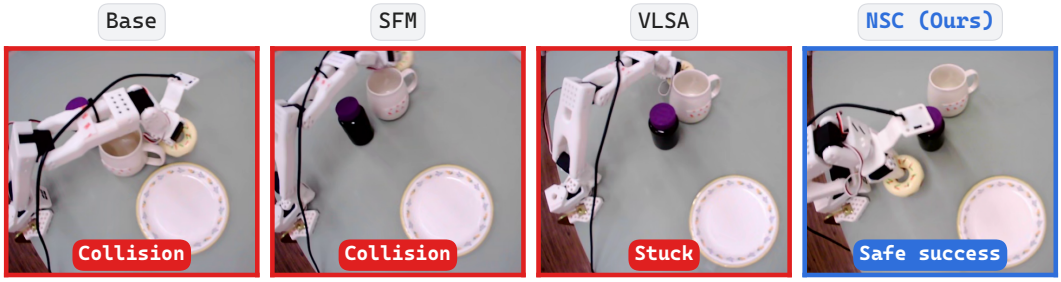


Figure 5: **Qualitative comparison on real-world experiments.**

collect 25 demonstrations for each route and use them to fine-tune $\pi_{0.5}$. During evaluation, we place an obstacle along the upper route. Figure 4 illustrates the experiment setup. We focus on episodes in which the base policy initially selects the unsafe upper route to assess whether each method can detour toward the safe lower route by correcting the current action to avoid a later collision. We report the performance with 10 episodes. Section B.5 details the experiment setup for real-world experiments.

Table 4 shows the results of the real-world experiments. When the base policy proposes an unsafe action chunk along the upper path, NSC corrects the entire action chunk toward the safe lower path learned by the base policy, enabling safe task completion. In contrast, SFM fails to avoid collisions. VLSA avoids collisions by enforcing action-level safety, but becomes stuck without finding a safe route to complete the task. We provide the qualitative comparison in Figure 5. These results demonstrate that NSC can be practically applied to real-world robot manipulation.

6 Conclusion

We presented NSC, a test-time safety filter that adjusts the initial noise of a frozen VLA policy using a control barrier function formulated in noise space. We derived a closed-form correction and established conditions for finite-step safety recovery. Experiments on SafeLIBERO, VLA-Arena, SafeDetour, and a real robot demonstrate improved task success while maintaining safety. These results support noise-space correction as an effective approach to integrating safety into action chunk generation without additional policy training.

Limitation and Future works. We are interested in extending our idea to ensure safety of VLA models. First, as the performance of NSC depends on the capability of base policy, applying our method into more sophisticated, closed-source models might be effective. Our evaluation also assumes that we know the ground truth obstacle geometry in order to isolate the effect of safety filters. Future work includes integrating accurate obstacle perception with estimation uncertainty.

AI USE STATEMENT

Large language models (LLMs) were employed exclusively for auxiliary purposes such as polishing the manuscript's grammar and assisting with basic code debugging and refactoring. LLMs were not involved in formulating the research questions, developing the core methodology, or designing the experiments. All underlying concepts, analytical methods, and findings remain the exclusive, original work of the authors.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained Policy Optimization. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 22–31. PMLR, 2017. URL https://proceedings.mlr.press/v70/achiam17a.html.

Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4): 181–185, 1985. doi: 10.1016/0020-0190(85)90056-0. URL https://doi.org/10.1016/0020-0190(85)90056-0.

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. doi: 10.23919/ECC.2019.8796030. URL https://doi.org/10.23919/ECC.2019.8796030.

Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1 – 3, 1966.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Remi Cadene, Simon Alibert, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, Dana Aubakirova, Mustafa Shukor, Jess Moss, Alexander Soare, Quentin Lhoest, Quentin Gallouédec, and Thomas Wolf. LeRobot: An Open-Source Library for End-to-End Robot Learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 122398–122417, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/c702175bc31584c52c28cc20731db371-Abstract-Conference.html.

Songqiao Hu, Zeyi Liu, Shuang Liu, Jun Cen, Zihan Meng, Shihefeng Wang, Xiang Li, and Xiao He. VLSA: Vision-Language-Action Models with Plug-and-Play Safety Constraint Layer. In *2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026. URL https://arxiv.org/abs/2512.11891. Accepted for publication.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_\{0.5\}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Rob Knight, Pepijn Kooijmans, Remi Cadene, Simon Alibert, Michel Aractingi, Dana Aubakirova, Adil Zouitine, Russi Martino, Steven Palma, Caroline Pascal, and Thomas Wolf. Standard Open SO-100 & SO-101 Arms. GitHub repository, 2024. URL https://github.com/TheRobotStudio/SO-ARM100.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations*, 2023. URL https://iclr.cc/virtual/2023/poster/11309.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-Scale Simulation of Household Tasks for Generalist Robots. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.050. URL https://www.roboticsproceedings.org/rss20/p050.html.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A Large-Scale Simulation Framework for Training and Benchmarking Generalist Robots. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/a05003fdb1e9562ab0c0a9719ea4de10-Abstract-Conference.html.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio G'omez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gim'enez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A Generalist Agent. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=1ikK0kHjvj.

Wei Xiao, Johnson (Tsun-Hsuan) Wang, Chuang Gan, Ramin Hasani, Mathias Lechner, and Daniela Rus. SafeDiffuser: Safe Planning with Diffusion Probabilistic Models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 100391–100412, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f95606d8e870020085990d9650b4f2a1-Abstract-Conference.html.

Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, Yingyan Li, Futeng Liu, Wanli Peng, Optimus Qin, Yifei Su, Nan Sun, Qiao Sun, Runze Suo, Heyun Wang, Yunhong Wang, Rujie Wu, Caoyu Xia, Lina Zhang, Jack Zhao, Guoliang Chen, Wenlong Chen, Xinze He, Bin Li, Qing Li, Zhuorong Li, Heng Qu, Wenxuan Song, Diyun Xiang, Yifan Xie, Peiran Xu, Hangjun Ye, Wen Ye, Han Zhao, and Quanyun Zhou. Xiaomi-Robotics-1: Scaling Vision-Language-Action Models with over 100K Hours of Real-World Trajectories. *arXiv preprint arXiv:2607.15330*, 2026. URL https://arxiv.org/abs/2607.15330.

Jeongyong Yang, Seunghwan Jang, and SooJean Han. SafeFlowMatcher: Safe and Fast Planning using Flow Matching with Control Barrier Functions. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 54058–54087, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/58b1b19bbbcce03bfb66ea60344e41ea-Abstract-Conference.html.

Borong Zhang, Yuhao Zhang, Jiaming Ji, Yingshan Lei, Juntao Dai, Yuanpei Chen, and Yaodong Yang. SafeVLA: Towards Safety Alignment of Vision-Language-Action Model via Constrained Learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 153335–153373. Curran Associates, Inc., 2025. doi: 10.52202/085713-5128. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/e185c7be603426028c32ae1003a59d78-Abstract-Conference.html.

Borong Zhang, Jiahao Li, Jiachen Shen, Yuhao Zhang, Yishuai Cai, Yuanpei Chen, Juntao Dai, Jiaming Ji, and Yaodong Yang. VLA-Arena: An Open-Source Framework for Benchmarking Vision-Language-Action Models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/61388.

A Theoretical Analysis

We use the notation of Section 4. The conditioning input s, safety function h, initial noise z_0, and margin $\delta > 0$ are fixed throughout the analysis. The assumptions on h and π imply that H is continuously differentiable.

A.1 Continuous-Time Safety Recovery

To specify the recovery neighborhood in Proposition 4.1, assume there exists a constant $c > 0$ such that g is locally Lipschitz and $\|g(z)\|_2 \geq c$ throughout an open region containing

$$\left\{ z \in \mathbb{R}^m : \|z - z_0\|_2 \leq -\frac{H(z_0)}{c}, \quad H(z_0) \leq H(z) \leq 0 \right\}. \tag{12}$$

This open region is the recovery neighborhood. Here c is a uniform positive lower bound on the safety gradient's magnitude throughout the region, ensuring that the gradient remains nonzero.

Proof of Proposition 4.1. In Equation 10, the denominator satisfies $\|g(z)\|_2^2 \geq c^2 > 0$, so division is well defined. Since H is continuously differentiable, g is locally Lipschitz, and $[v]_+ = \max\{v, 0\}$ is Lipschitz continuous, $u^\star$ is locally Lipschitz on the recovery neighborhood. Hence $dz/d\tau = u^\star(z)$ with $z(0) = z_0$ has a unique solution for a short interval of optimization time. We next show that the solution can be extended until it reaches $H = 0$.

While $H(z) < 0$, the chain rule and the CBF constraint give $dH/d\tau = g(z)^\top u^\star(z) \geq \delta - H(z) > 0$. Thus H increases strictly, and $z(\tau) \neq z_0$ for every positive τ before safety is reached. We now bound how far the noise can move from z_0. In the decomposition of Equation 10, the component orthogonal to $g(z)$ has inner product $-\|P_{g^\perp}(z - z_0)\|_2^2 \leq 0$ with $z - z_0$. Applying the Cauchy–Schwarz inequality to the component along $g(z)$ therefore gives, for positive τ before safety is reached,

$$\begin{aligned} \frac{d}{d\tau}\|z - z_0\|_2 &= \frac{(z - z_0)^\top u^\star(z)}{\|z - z_0\|_2} \\ &\leq \frac{g(z)^\top u^\star(z)}{\|g(z)\|_2} \leq \frac{1}{c}\frac{dH}{d\tau}. \end{aligned} \tag{13}$$

Integrating this inequality from a positive optimization time to τ and then letting the starting time approach zero gives, by continuity and $z(0) = z_0$,

$$\|z(\tau) - z_0\|_2 \leq \frac{H(z(\tau)) - H(z_0)}{c} \leq -\frac{H(z_0)}{c}. \tag{14}$$

Before reaching safety, the noise stays in the closed, bounded set in Equation 12, where continuity makes $u^\star$ bounded. The noise therefore moves at a bounded speed and has a limit in this set at any finite endpoint of its time interval. Because this set lies inside the recovery neighborhood, the differential equation has a local solution from that limit point. The solution can therefore be extended until it reaches $H = 0$.

Finally, multiplying $dH/d\tau + H \geq \delta$ by e^τ and integrating gives

$$H(z(\tau)) \geq \delta + \big(H(z_0) - \delta\big)e^{-\tau}. \tag{15}$$

This lower bound on the safety score reaches zero at $\tau = \log(1 - H(z_0)/\delta)$. The solution exists up to this time unless safety is reached earlier, so $\tau_{\text{hit}} \leq \log(1 - H(z_0)/\delta)$. By continuity, Equation 15 also holds at $\tau = \tau_{\text{hit}}$. □

A.2 Discrete-Time Safety Recovery

Let z_k denote the noise after k updates, starting from z_0. We analyze $z_{k+1} = z_k + \eta u^\star(z_k)$ with a fixed step size $\eta > 0$. The following assumptions apply to this discrete sequence independently of the continuous-time assumptions.

Proposition A.1 (Finite-step safety recovery). *Suppose $H(z_0) < 0$, and let K be the maximum number of noise updates, a nonnegative integer. At every generated z_k with $k < K$ and $H(z_k) < 0$, assume $\|g(z_k)\|_2 \geq c > 0$, where c is a fixed lower bound on the gradient norm. For each such update, assume that g is Lipschitz continuous along the straight path from z_k to z_{k+1} in noise space: the Euclidean norm of the gradient difference between any two points is at most $L > 0$ times their distance. The constants c and L apply to all these updates, including the update that first reaches safety. We write $\lceil \cdot \rceil$ for rounding up to the nearest integer. If the fixed step size η and budget K satisfy*

$$\begin{aligned} 0 < \eta \leq \min\left\{1,\ \frac{\delta c^2}{L\big((\delta - H(z_0))^2 + 4H(z_0)^2\big)}\right\}, \\ K \geq \left\lceil \frac{\log\big(1 - H(z_0)/\delta\big)}{-\log(1-\eta/2)} \right\rceil, \end{aligned} \tag{16}$$

then Algorithm 1 returns a chunk satisfying $h(A) \geq 0$ within K noise updates.

Proof. At an unsafe noise $z = z_k$ with $k < K$, the condition $\|g(z)\|_2 \geq c > 0$ ensures that the QP has a solution. When the CBF constraint holds with equality, so that $g(z)^\top u^\star(z) = \delta - H(z)$, Equation 10 gives a component along $g(z)$ with norm $(\delta - H(z))/\|g(z)\|_2$ and an orthogonal component with norm at most $\|z - z_0\|_2$. Their squared norms add because the two components are orthogonal. When the constraint holds with strict inequality, the QP returns the reference direction $u^\star(z) = -(z - z_0)$. In both cases,

$$\|u^\star(z)\|_2^2 \leq \frac{(\delta - H(z))^2}{\|g(z)\|_2^2} + \|z - z_0\|_2^2. \tag{17}$$

For $0 < \eta \leq 1$, Equation 10, the triangle inequality, and $[\delta - H(z) + g(z)^\top(z - z_0)]_+ \leq \delta - H(z) + \|g(z)\|_2\|z - z_0\|_2$ imply

$$\|z + \eta u^\star(z) - z_0\|_2 \leq \|z - z_0\|_2 + \frac{\eta(\delta - H(z))}{c}. \tag{18}$$

We prove by induction that every generated z_k with $H(z_k) < 0$ has a safety score at least as large as the initial score and satisfies the distance bound

$$H(z_0) \leq H(z_k) < 0, \qquad \|z_k - z_0\|_2 \leq \frac{2\big(H(z_k) - H(z_0)\big)}{c}. \tag{19}$$

These inequalities hold at $k = 0$. Assuming they hold at $k < K$, Equation 17 yields $\|u^\star(z_k)\|_2^2 \leq \big((\delta - H(z_0))^2 + 4H(z_0)^2\big)/c^2$. Consequently, the step-size condition in Equation 16 ensures $L\eta\|u^\star(z_k)\|_2^2 \leq \delta \leq g(z_k)^\top u^\star(z_k)$. The condition on L bounds the error in the first-order approximation of H from z_k to z_{k+1}, giving

$$\begin{aligned} H(z_{k+1}) - H(z_k) &\geq \eta g(z_k)^\top u^\star(z_k) - \frac{L\eta^2}{2}\|u^\star(z_k)\|_2^2 \\ &\geq \frac{\eta}{2}\big(\delta - H(z_k)\big). \end{aligned} \tag{20}$$

If $H(z_{k+1}) \geq 0$, the algorithm returns the generated chunk $\pi(z_{k+1}, s)$. Otherwise, its safety score increases, and Equation 18 and Equation 20 give

$$\begin{aligned} \|z_{k+1} - z_0\|_2 &\leq \|z_k - z_0\|_2 + \frac{2\big(H(z_{k+1}) - H(z_k)\big)}{c} \\ &\leq \frac{2\big(H(z_{k+1}) - H(z_0)\big)}{c}. \end{aligned} \tag{21}$$

This proves Equation 19 at $k + 1$.

Applying Equation 20 successively from z_0 gives

$$H(z_k) \geq \delta + \big(H(z_0) - \delta\big)(1 - \eta/2)^k \tag{22}$$

for all generated iterates, including the first one with $H(z_k) \geq 0$. If no earlier iterate is safe, this bound applies at $k = K$ and gives $H(z_K) \geq 0$ under Equation 16. Algorithm 1 checks this final candidate before declaring failure, so it returns a safe chunk within K updates. □

B Experimental Details

B.1 Common Implementation Details

Safety specification. The safety function $h(A)$ checks the motion that a chunk A is predicted to produce against the obstacles in the scene. We predict the positions of the arm links and the gripper over the chunk by accumulating its commanded end-effector translations from the current state, and obtain the obstacle geometry at every query, holding it fixed over the chunk. With $d_k(A)$ the smallest signed clearance between the robot and the obstacles at step k and a margin ϵ (Table 6),

$$h(A) = \min_{k \le T} d_k(A) - \epsilon,$$

so $h(A) \ge 0$ certifies a clearance of at least ϵ over the whole chunk. Replacing each minimum with a softmin gives a smooth lower bound $\tilde{h} \le h$.

Evaluation metrics. The task success rate (TSR) is the percentage of episodes that complete the task within the horizon, and the collision avoidance rate (CAR) is the percentage of collision-free episodes. The safe success rate (SS) is the percentage of episodes that are both collision-free and successful.

B.2 Toy Experiments

Environment. The toy task is 2D navigation from a fixed start to a goal region. An obstacle blocks the right half of the field, so the only way to reach the goal safely is to take the left path. A flow-matching MLP is trained without obstacles on a route that is symmetric to the center line, proposing left and right routes equally. Every right-going chunk becomes unsafe once the obstacles are added. Each chunk spans the whole episode, so a single query fixes the route, and the safety function, certifing the entire trajectory.

Details of evaluation. All filters correct the same 200 noise draws, so the methods differ only in the correction. A trial is safe if its executed path never touches an obstacle or wall and succeeds if it also reaches the goal. SafeFlowMatcher corrects each action independently, so its paths change abruptly within the chunk. VLSA does not collide to the obstacle, but many of them can not reach the goal since it stalls below the obstacles. NSC instead regenerates each unsafe chunk through the policy, so its paths take the open left route from the start.

B.3 VLA Safety Benchmarks

B.3.1 SafeLIBERO

Task. SafeLIBERO (Hu et al., 2026) adds obstacles to four tasks from each of the four LIBERO suites (Spatial, Object, Goal, and Long). Each task has two safety levels. **Level I**: obstacles are placed near the target object. **Level II**: obstacles directly obstruct the arm's movement path. We evaluate on this harder Level II. Everyday objects, such as moka pots, storage boxes, milk cartons, wine bottles, mugs, and books, are randomly sampled across the 50 episodes of each task.

Policy. We use the public openpi checkpoint `pi05_libero` for $\pi_{0.5}$ (Intelligence et al., 2025) and NVIDIA's `GR00T-N1.7-LIBERO` release for GR00T N1.7 (Bjorck et al., 2025). Observations (two camera views, the end-effector and gripper state, and the task instruction) are fed into both policies. Both policies output delta end-effector poses, executed by robosuite's OSC controller at 20 Hz. $\pi_{0.5}$ integrates its flow in 10 steps into a chunk of $T = 10$ actions and executes the first $T_{\text{exec}} = 5$. GR00T N1.7 uses 4 denoising steps with $T = 16$ and $T_{\text{exec}} = 8$. VLSA, NSC filters share these same settings and for SafeFlowMatcher, the details are listed in Table 6.

Details of evaluation. Each run evaluates the 50 episodes per task, and the three runs differ only in the seed of the policy's sampling noise. An episode runs until success or for at most 300 steps (550 for Long). When an obstacle moves more than 1 mm from its settled pose, a collision is recorded. Each metric is computed per run over the episodes of a task, a suite. Then over the three runs, we report the mean and sample standard deviation.

B.3.2 VLA-ARENA

Task. We use the three Safety suites of VLA-Arena (Zhang et al., 2026). Each suite has five tasks at each difficulty level. **StaticObstacles**: the robot picks up a fruit or vegetable and places it on a plate or in a bowl. The workspace has zero (L0), one (L1) or two (L2) fragile obstacles, such as mugs and bottles, standing on the path. **HazardAvoidance**: the robot places an object in a bowl or on a plate next to an active hazard, a lit candle or a turned-on stove, and must keep the gripper and the object away from it. The hazard lies away from the path in L0, next to it in L1, and on the direct route in L2. **DynamicObstacles**: the robot pushes an object into the region between two mugs or teapots, or places an apple in a bowl, among toy obstacles. The obstacles are stationary in L0, one moves linearly across the path in L1, and in L2 additional obstacles follow curved trajectories.

Policy. We use the $\pi_{0.5}$ checkpoint released with the benchmark without further fine-tuning. It integrates its flow in 10 Euler steps into a chunk of $T = 10$ actions and executes the first $T_{\text{exec}} = 5$. Each action is a delta end-effector pose with a gripper command, executed by robosuite's OSC controller at 20 Hz. Settings are shared by the base policy, VLSA, and NSC. SafeFlowMatcher's integration steps are listed in Table 6.

Details of evaluation. We evaluate the L1 and L2 tasks of the three suites (30 tasks). Episode k of a task starts from its k-th fixed initial state for every run, with *TBD* episodes per task. An episode runs for at most 300 policy steps. The benchmark's task-completion predicate determines task success, and an episode counts as collision-free if it incurs no cost (Zhang et al., 2026). The three runs differ only in their seed, setting the policy's sampling noise. Each metric is computed per run over the episodes, and the reported mean and sample standard deviation are taken over the three runs.

B.4 SAFEDETOUR

Design. SafeDetour tests whether a filter can anticipate hazards later in the action chunk. Each scene offers two routes that complete the task (Figure 3): (i) a *trap route*, which the policy prefers and which is collision-free at first but leads into a hazard, so a filter that only avoids immediate collisions ends in a collision or a deadlock; and (ii) a *detour route*, which is collision-free throughout. Each scene starts from a procedurally generated RoboCasa kitchen, with layout and style drawn from a seed. We roll out the base policy in this kitchen without hazards and place the primary hazard on the dominant mode of its trajectories, so that the route the policy prefers becomes the trap route. Hazard objects are sized to fit their fixture and tall enough to intersect this route: glasses and glass bottles that fit inside the cabinet, a kettle and a pot on the stove burners, and wine bottles that obstruct the arm as it closes the toaster-oven door. A candidate is discarded if its objects start in collision, do not settle, or leave no collision-free route to task completion. We draw 30 scenes per task and evaluate 5 episodes for each scene.

Task. We instantiate SafeDetour on the three tasks shown in Figure 3, taken from the Atomic-Seen target tasks of RoboCasa365 (Nasiriany et al., 2026): `PickPlaceCounterToCabinet` (Cabinet), `PickPlaceCounterToStove` (Stove), and `CloseToasterOvenDoor` (Toaster). We chose them because obstacles placed in their scenes readily create both a trap route and a detour route, and because the base policy already solves them with a high success rate. **Cabinet**: the robot places an object in a cabinet. Glasses stand at the front of the cabinet, and a wine-glass trap sits at the policy's usual placement spot; (i) the trap route follows the usual placement path, where avoiding only the glasses ahead leads into the trap, and (ii) the detour route places the object at another free spot. **Stove**: the robot places an object in a pan on a stove. The pan sits on a rear burner, and a kettle and a pot stand staggered on the front burners; (i) the trap route crosses the front burners, where avoiding only the nearer vessel drives the arm into the other, and (ii) the detour route leaves the usual path from the start, moving toward the rear or upward. **Toaster**: the robot closes an open toaster-oven door, which the policy usually lifts at its left or right corner. One bottle stands at a corner, and another stands behind the path to the opposite corner; (i) the trap route switches to the opposite corner and hits the second bottle, and (ii) the detour route lifts the door toward its middle instead of at either corner. Each setting tests whether the detour becomes preferable once safety is evaluated over the whole chunk.

Policy. All filters share the Xiaomi-Robotics-1 (Xiaomi Robotics Team et al., 2026) checkpoint, a 5B-parameter flow-matching VLA. Each query integrates the flow ODE from Gaussian noise with five Euler steps, producing a chunk of $T = 16$ actions whose end-effector pose deltas are tracked by an OSC controller at 20 Hz. All $T_{\text{exec}} = 16$ actions are executed before the next query. Every episode starts with the same noise from the same seed.

B.5 Real-World Experiments

Hardware. We use the SO101 arm (Knight et al., 2024) with two Sunplus SPCA2281 webcams: a wrist-mounted camera and a fixed camera providing a top-down view of the workspace. Both cameras capture RGB images at 640×480 resolution and 30 Hz.

Design. We design a **PickandPlace** task in which the robot moves a donut to a plate via either an upper or a lower route around a central obstacle (plastic bottle). At evaluation, we place an additional obstacle (mug) along the upper route, making the lower route the only collision-free path to task completion. We localize the mug by hand-guiding the gripper fingertip to eight rim points and the table surface, using forward kinematics to obtain their coordinates in the robot base frame. The estimated center, radius, and height then parameterize the obstacle geometry in the safety function. We illustrate the setup in Figure 4.

Policy. We collect 50 demonstrations through leader-follower teleoperation, with 25 demonstrations for each of the upper and lower routes. We fine-tune $\pi_{0.5}$ on this dataset for 2,500 training steps. We use the pre-trained checkpoints and fine-tuning recipe provided by LeRobot (Cadene et al., 2026).

Details of evaluation. We evaluate only episodes in which the base policy proposes an action chunk along the upper route, which is unsafe at test time. This setting captures a practical need for safety filters: correcting the base policy's proposed action when a newly introduced obstacle makes its intended path unsafe. We exclude episodes with action along the lower route, as they already follow a collision-free safe path and require no detour. This protocol tests whether each method can redirect the robot toward the safe lower route while still completing the task. We report the performance with 10 episodes.

B.6 Hyperparameters

Hyperparameter selection. For NSC, the safety margin ε, measured in meters, specifies the required clearance from obstacles through the condition $h(A) \geq \varepsilon$. As described in Algorithm 1, the noise correction also uses a step size η, a CBF margin δ, and an update budget K. We select η adaptively through an Armijo line search (Armijo, 1966) and fix $K = 10$ in all experiments. For VLSA (Hu et al., 2026), α is the gain in the linear CBF condition $\dot{h} + \alpha h \geq 0$, controlling the allowable rate of decrease in the safety function. We use the original paper's default value of $\alpha = 10$ across all tasks. For SafeFlowMatcher (Yang et al., 2026), T^p and T^c denote the numbers of prediction and correction steps, respectively, while α scales the flow vector field during correction. We require $T^p + T^c = \text{NFE}$, where NFE denotes each base policy's fixed number of flow-model evaluations. Table 5 summarizes the sweep ranges. The selected hyperparameter values for each benchmark and task are reported in Table 6.

Table 5: Hyperparameter sweep ranges. The NSC safety margin is swept over all tasks.

Method	Base policy	Parameter	Values
NSC	All	ε (m)	$\{0, 0.01, 0.02, 0.03\}$
SFM	All	α	$\{1.0, 1.5, 2.0, 2.5, 3.0\}$
	$\pi_{0.5}$	(T^c, T^p)	$\{(1,9), (2,8), (4,6), (8,2)\}$
	GR00T N1.7	(T^c, T^p)	$\{(1,3), (2,2), (3,1)\}$
	Xiaomi-Robotics-1	(T^c, T^p)	$\{(1,4), (2,3), (4,1)\}$

Table 6: Hyperparameters for each benchmark and task. ϵ is the safety margin. For SFM, T^p and T^c denote the numbers of prediction and correction steps, respectively, and α is the time-scaling constant.

	NSC	SFM		
Benchmark / task	ϵ	T^p	T^c	α
Toy				
2D navigation	0	2	8	2
SafeLIBERO ($\pi_{0.5}$)				
Spatial	0.03	8	2	2
Object	0.01	1	9	1
Goal	0.01	2	8	2
Long	0.01	2	8	3
SafeLIBERO (GR00T N1.7)				
Spatial	0.01	2	2	2
Object	0.01	2	2	2
Goal	0.03	1	3	2
Long	0.01	1	3	2
VLA-Arena				
StaticObstacles	0	4	6	2
DynamicObstacles	0	4	6	2
HazardAvoidance	0	2	8	2
SafeDetour				
Cabinet	0	1	4	2
Stove	0	1	4	2
Toaster	0	1	4	2
Real world				
Pick and Place	0	2	8	2

B.7 Computational Resources

Computing infrastructure. All experiments were run on two nodes, each with eight NVIDIA GeForce RTX 3090 GPUs (24 GB) under Ubuntu 24.04.

C Control Barrier Function Quadratic Programs

We derive the closed-form solution for a single scalar CBF constraint with unrestricted control inputs. Fix a state $x \in \mathcal{D} \subseteq \mathbb{R}^{d_x}$, where $\mathcal{D}$ is the open state domain and d_x is the state dimension. The control input is $u \in \mathbb{R}^{d_u}$, where d_u is the input dimension. In the dynamics $\dot{x} = f(x) + G(x)u$, the drift vector is $f(x) \in \mathbb{R}^{d_x}$ and the control matrix is $G(x) \in \mathbb{R}^{d_x \times d_u}$. We use the continuously differentiable safety function $h : \mathcal{D} \to \mathbb{R}$ and the extended class-$\mathcal{K}$ function α from the preliminaries. The domain of α contains all values of h on $\mathcal{D}$. The vector $\nabla h(x) \in \mathbb{R}^{d_x}$ is the gradient of h with respect to x, and $\|\cdot\|_2$ denotes the Euclidean norm. Let $u_{\mathrm{ref}}(x) \in \mathbb{R}^{d_u}$ be the reference input. All quantities evaluated at x are fixed when optimizing over u.

Whenever the CBF-QP is feasible, its minimizer is denoted by $u^\star(x)$ and is defined by

$$\begin{aligned} u^\star(x) = \underset{u \in \mathbb{R}^{d_u}}{\arg\min} \quad & \frac{1}{2}\|u - u_{\mathrm{ref}}(x)\|_2^2 \\ \text{subject to} \quad & \nabla h(x)^\top \big(f(x) + G(x)u\big) + \alpha\big(h(x)\big) \geq 0. \end{aligned} \tag{23}$$

Define the scalar

$$\psi(x) = \nabla h(x)^\top \big(f(x) + G(x)u_{\mathrm{ref}}(x)\big) + \alpha\big(h(x)\big). \tag{24}$$

This is the left-hand side of the CBF constraint evaluated at the reference input, so the reference is feasible exactly when $\psi(x) \geq 0$.

Existence and uniqueness. First suppose that $G(x)^\top \nabla h(x) \neq 0$. The CBF constraint defines a nonempty closed half-space in control space. In particular, adding a sufficiently large positive multiple of $G(x)^\top \nabla h(x)$ to any input makes the constraint strict. The objective is continuous, tends to infinity as $\|u\|_2 \to \infty$, and is strictly convex. It therefore attains a unique minimum on the feasible set. The nonzero condition is maintained throughout the derivation below, and the zero case is considered at the end.

Optimality conditions. Introduce a scalar Lagrange multiplier $\lambda \geq 0$. Because the CBF constraint is written with a nonnegative left-hand side, the Lagrangian is

$$\begin{aligned} \mathcal{L}(u, \lambda) = & \frac{1}{2}\|u - u_{\mathrm{ref}}(x)\|_2^2 \\ & - \lambda \left[\nabla h(x)^\top \big(f(x) + G(x)u\big) + \alpha\big(h(x)\big)\right]. \end{aligned} \tag{25}$$

The problem is convex and admits a strictly feasible input, so the Karush–Kuhn–Tucker (KKT) conditions are necessary and sufficient for optimality. Writing $\lambda^\star$ for the optimal multiplier, these conditions are

$$\begin{aligned} u^\star(x) - u_{\mathrm{ref}}(x) - \lambda^\star G(x)^\top \nabla h(x) &= 0, \\ \nabla h(x)^\top \big(f(x) + G(x)u^\star(x)\big) + \alpha\big(h(x)\big) &\geq 0, \\ \lambda^\star &\geq 0, \\ \lambda^\star \left[\nabla h(x)^\top \big(f(x) + G(x)u^\star(x)\big) + \alpha\big(h(x)\big)\right] &= 0. \end{aligned} \tag{26}$$

The four conditions express stationarity with respect to u, satisfaction of the CBF constraint, nonnegativity of the multiplier, and complementary slackness, respectively. The stationarity condition gives

$$u^\star(x) = u_{\mathrm{ref}}(x) + \lambda^\star G(x)^\top \nabla h(x). \tag{27}$$

Substituting this expression into the CBF constraint and using the definition of $\psi(x)$ yields

$$\begin{aligned} & \nabla h(x)^\top \big(f(x) + G(x)u^\star(x)\big) + \alpha\big(h(x)\big) \\ & \quad = \psi(x) + \lambda^\star \|G(x)^\top \nabla h(x)\|_2^2. \end{aligned} \tag{28}$$

Solving for the multiplier. If $\psi(x) \geq 0$, the reference input is feasible and achieves the smallest possible objective value, zero. Hence $u^\star(x) = u_{\mathrm{ref}}(x)$ and $\lambda^\star = 0$. This includes $\psi(x) = 0$, where the constraint holds with equality but no correction is needed.

If $\psi(x) < 0$, setting $\lambda^\star = 0$ would return the infeasible reference input. Thus $\lambda^\star > 0$, and complementary slackness requires the CBF constraint to hold with equality. Consequently,

$$\psi(x) + \lambda^\star \|G(x)^\top \nabla h(x)\|_2^2 = 0, \qquad \lambda^\star = \frac{-\psi(x)}{\|G(x)^\top \nabla h(x)\|_2^2}. \tag{29}$$

Combining the two cases gives $\lambda^\star = [-\psi(x)]_+ / \|G(x)^\top \nabla h(x)\|_2^2$, where $[v]_+ = \max\{v, 0\}$ for a real scalar v. Substitution into the stationarity condition gives the unique closed-form solution

$$u^\star(x) = u_{\mathrm{ref}}(x) + \frac{[-\psi(x)]_+}{\|G(x)^\top \nabla h(x)\|_2^2} G(x)^\top \nabla h(x). \tag{30}$$

The solution is the Euclidean projection of the reference input onto the feasible half-space. When the reference violates the constraint, the correction points along $G(x)^\top \nabla h(x)$, the normal to the constraint boundary directed into the feasible half-space.

Zero coefficient of the control input. If $G(x)^\top \nabla h(x) = 0$, the CBF constraint is independent of u and reduces to

$$\nabla h(x)^\top f(x) + \alpha\big(h(x)\big) \geq 0. \tag{31}$$

If this inequality holds, every input is feasible and the unique minimizer is $u^\star(x) = u_{\mathrm{ref}}(x)$. Otherwise, the QP is infeasible. If h satisfies the CBF condition on $\mathcal{D}$ with this α and unrestricted inputs, this infeasible case cannot occur.

The derivation concerns the optimizer at a fixed state for one CBF constraint with unrestricted controls. Forward invariance additionally requires the assumptions given in the preliminaries.

C.1 Closed-Form Solution of the Noise-Space CBF-QP

Let m denote the noise dimension. Fix the current noise $z \in \mathbb{R}^m$, the initial noise $z_0 \in \mathbb{R}^m$, and the conditioning input s. Let π be the frozen policy and h the chunk safety function. The scalar $H(z) = h(\pi(z, s))$ is the safety score of the generated chunk. Assume that H is differentiable at the current noise z, as implied by the differentiability assumptions, and define $g(z) = \nabla_z H(z) \in \mathbb{R}^m$. For the CBF parameter $\delta > 0$, NSC selects the noise-update direction $u \in \mathbb{R}^m$ by solving

$$\begin{aligned} \min_{u \in \mathbb{R}^m} \quad & \frac{1}{2}\|u + (z - z_0)\|_2^2 \\ \text{subject to} \quad & \delta - H(z) - g(z)^\top u \leq 0. \end{aligned} \tag{32}$$

Here $\|\cdot\|_2$ denotes the Euclidean norm, and $-(z - z_0)$ is the reference direction toward the initial noise. Throughout this derivation, $H(z)$ and $g(z)$ are fixed when optimizing over u. We first consider $g(z) \neq 0$. The zero-gradient case is discussed at the end of this subsection.

Existence and uniqueness. A sufficiently large positive multiple of $g(z)$ satisfies the constraint strictly, so the feasible set is a nonempty closed halfspace. The objective is continuous and tends to infinity as $\|u\|_2$ tends to infinity, so it attains a minimum on this feasible set. Its Hessian with respect to u is I_m, the $m \times m$ identity matrix, so the objective is strictly convex and the minimizer $u^\star(z)$ is unique. Convexity and strict feasibility make the Karush–Kuhn–Tucker (KKT) conditions necessary and sufficient for optimality.

Lagrangian and optimality conditions. Introduce a scalar Lagrange multiplier $\lambda \geq 0$ for the inequality constraint and define

$$\mathcal{L}(u, \lambda) = \frac{1}{2}\|u + (z - z_0)\|_2^2 + \lambda\big(\delta - H(z) - g(z)^\top u\big). \tag{33}$$

Let $\lambda^\star$ denote the multiplier at the optimum. The KKT conditions are

$$\begin{aligned} u^\star(z) + (z - z_0) - \lambda^\star g(z) = 0, \quad & \text{(stationarity)} \\ \delta - H(z) - g(z)^\top u^\star(z) \leq 0, \quad & \text{(primal feasibility)} \\ \lambda^\star \geq 0, \quad & \text{(dual feasibility)} \\ \lambda^\star\big(\delta - H(z) - g(z)^\top u^\star(z)\big) = 0, \quad & \text{(complementary slackness).} \end{aligned} \tag{34}$$

Stationarity gives

$$u^\star(z) = -(z - z_0) + \lambda^\star g(z). \tag{35}$$

Thus any additional correction to the reference direction is parallel to $g(z)$.

Determining the multiplier. If the reference direction already satisfies the constraint, then

$$-g(z)^\top (z - z_0) \geq \delta - H(z). \tag{36}$$

Choosing $u^\star(z) = -(z - z_0)$ and $\lambda^\star = 0$ satisfies all KKT conditions. The objective is also zero at this direction, its smallest possible value. This case includes equality in equation 36, where the constraint holds with equality but the multiplier is still zero.

Otherwise, the reference direction violates the constraint, so $-g(z)^\top (z - z_0) < \delta - H(z)$. The multiplier must then be strictly positive, since $\lambda^\star = 0$ would return the infeasible reference direction. Complementary slackness therefore requires $g(z)^\top u^\star(z) = \delta - H(z)$. Substituting equation 35 yields

$$\begin{aligned} -g(z)^\top (z - z_0) + \lambda^\star \|g(z)\|_2^2 &= \delta - H(z), \\ \lambda^\star &= \frac{\delta - H(z) + g(z)^\top (z - z_0)}{\|g(z)\|_2^2} > 0. \end{aligned} \tag{37}$$

Combining the two cases gives

$$\lambda^\star = \frac{\max\{0, \delta - H(z) + g(z)^\top (z - z_0)\}}{\|g(z)\|_2^2}. \tag{38}$$

Substitution into equation 35 gives the closed-form solution

$$u^\star(z) = -(z - z_0) + \frac{\max\{0, \delta - H(z) + g(z)^\top (z - z_0)\}}{\|g(z)\|_2^2}\, g(z). \tag{39}$$

Orthogonal decomposition. Taking the inner product of equation 39 with $g(z)$ gives

$$\begin{aligned} g(z)^\top u^\star(z) &= -g(z)^\top (z - z_0) + \max\{0, \delta - H(z) + g(z)^\top (z - z_0)\} \\ &= \max\{\delta - H(z), -g(z)^\top (z - z_0)\}. \end{aligned} \tag{40}$$

Define the orthogonal projector onto the subspace perpendicular to $g(z)$ by

$$P_{g^\perp} = I_m - \frac{g(z) g(z)^\top}{\|g(z)\|_2^2}. \tag{41}$$

Since $P_{g^\perp} g(z) = 0$, applying this projector to equation 35 gives $P_{g^\perp} u^\star(z) = -P_{g^\perp}(z - z_0)$. Splitting the solution into its parallel and perpendicular components therefore yields

$$\begin{aligned} u^\star(z) &= \frac{g(z)^\top u^\star(z)}{\|g(z)\|_2^2}\, g(z) - P_{g^\perp}(z - z_0) \\ &= \frac{\max\{\delta - H(z), -g(z)^\top (z - z_0)\}}{\|g(z)\|_2^2}\, g(z) - P_{g^\perp}(z - z_0). \end{aligned} \tag{42}$$

The parallel component supplies the directional derivative $g(z)^\top u^\star(z) \geq \delta - H(z)$ required by the CBF constraint. The perpendicular component has zero inner product with $g(z)$ and hence contributes nothing to the first-order change of H. It pulls toward z_0 within the subspace perpendicular to $g(z)$, as quantified below. Since $P_{g^\perp}^\top = P_{g^\perp}$ and $P_{g^\perp}^2 = P_{g^\perp}$,

$$(z - z_0)^\top \left[-P_{g^\perp}(z - z_0)\right] = -\|P_{g^\perp}(z - z_0)\|_2^2 \leq 0. \tag{43}$$

Zero gradient and scope. If $g(z) = 0$, the constraint reduces to $0 \geq \delta - H(z)$. The QP is therefore infeasible when $H(z) < \delta$; when $H(z) \geq \delta$, its unique solution is the reference direction $-(z - z_0)$. NSC invokes the correction only for an unsafe chunk, for which $H(z) < 0$. Since $\delta > 0$, a zero gradient at such a candidate makes the QP infeasible, and the algorithm reports failure. The derivation above establishes the CBF condition at the current noise; safety after a finite noise update is checked by evaluating the regenerated chunk.

D Flow Matching Action Policies

We describe the linear-interpolation form of conditional flow matching (CFM) used for action generation (Lipman et al., 2023; Black et al., 2024). Let $s \in \mathcal{S}$ denote the policy input, including observations, robot state, and a language instruction, where $\mathcal{S}$ is the input space. An action chunk contains T actions $a_t \in \mathbb{R}^n$, where n is the dimension of each action and $t = 1, \dots, T$. The stacked chunk is $A = (a_1^\top, \dots, a_T^\top)^\top \in \mathbb{R}^{Tn}$.

Action representation. The flow model operates on vectors in $\mathbb{R}^m$, where m is the noise dimension. Let $y \in \mathbb{R}^m$ denote a demonstration chunk in the representation used to train the flow model, which may include normalization and padding. Let $D_s : \mathbb{R}^m \to \mathbb{R}^{Tn}$ denote the policy's output transformation for input s, such as removing padding and undoing normalization. Thus, the internal flow representation and the returned action chunk need not have the same dimension.

Flow matching objective. For a demonstration pair (s, y), sample independent Gaussian noise $z \sim \mathcal{N}(0, I_m)$, where I_m is the $m \times m$ identity matrix. Let $\xi \in [0, 1]$ denote internal flow sampling time. For fixed samples z and y, the interpolated training sample $\widetilde{X}_\xi \in \mathbb{R}^m$ and its velocity are

$$\widetilde{X}_\xi = (1 - \xi)z + \xi y, \qquad \frac{d\widetilde{X}_\xi}{d\xi} = y - z. \tag{44}$$

This interpolation is the zero-terminal-noise limit of the conditional Gaussian path in Lipman et al. (2023), whose original formulation retains a small positive Gaussian standard deviation at the endpoint. The velocity field conditioned on the policy input s, $v_\theta(\cdot, \xi; s) : \mathbb{R}^m \to \mathbb{R}^m$, with trainable parameters θ, is trained to predict this velocity by minimizing the conditional flow matching loss $\mathcal{L}_{\mathrm{CFM}}(\theta)$, the expected squared Euclidean error between the predicted and target velocities.

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{(s,y),z,\xi}\left[\left\| v_\theta(\widetilde{X}_\xi, \xi; s) - (y - z) \right\|_2^2\right]. \tag{45}$$

The expectation is over demonstration pairs (s, y) drawn from the training data, Gaussian noise z independent of (s, y), and flow times ξ sampled independently of both. The original CFM objective uses uniformly sampled flow times (Lipman et al., 2023). For a pretrained action policy, the distribution of ξ follows its training procedure, which may assign different weights to different flow times. The demonstration representation y supplies the training target and is not an input to v_θ. Both $\widetilde{X}_\xi$ and its target velocity are computed directly from the sampled noise and demonstration, so evaluating this loss does not require solving an ordinary differential equation (ODE).

Action generation. At inference, the base policy samples $z \sim \mathcal{N}(0, I_m)$ and evolves an intermediate sample $X_\xi \in \mathbb{R}^m$ according to

$$\frac{dX_\xi}{d\xi} = v_\theta(X_\xi, \xi; s), \qquad X_0 = z. \tag{46}$$

For this continuous-time description, we assume that the ODE has a unique solution on $[0, 1]$ for the inputs under consideration. The resulting path X_ξ is determined by the learned field and is generally different from an individual training interpolation $\widetilde{X}_\xi$. The velocity field acts on the representation of the entire chunk. Flow sampling time ξ indexes the generation process, not the execution of individual actions.

In practice, the policy uses a numerical approximation to the ODE. For example, let N be a positive integer specifying the number of forward Euler steps, and let $\widehat{X}_j$ approximate $X_{j/N}$ after j steps. Starting from $\widehat{X}_0 = z$, the update for $j = 0, \dots, N - 1$ is

$$\widehat{X}_{j+1} = \widehat{X}_j + \frac{1}{N} v_\theta\left(\widehat{X}_j, \frac{j}{N}; s\right). \tag{47}$$

For this sampler, the complete mapping from noise to the action chunk is

$$A = \pi(z, s) = D_s(\widehat{X}_N). \tag{48}$$

Thus, $\pi : \mathbb{R}^m \times \mathcal{S} \to \mathbb{R}^{Tn}$ includes both numerical sampling and output transformation. The N flow integration steps occur within one policy query and are distinct from the maximum number K of NSC noise updates.

Connection to noise-space correction. With the policy parameters, conditioning input, and deterministic sampling procedure fixed, the initial noise determines the output chunk. For fixed s, if $v_\theta(\cdot, j/N; s)$ for $j = 0, \ldots, N-1$ and D_s are continuously differentiable, then every Euler update and the output transformation are continuously differentiable in their vector inputs. Their finite composition therefore makes the implemented policy $\pi(\cdot, s)$ continuously differentiable in z. For the continuously differentiable chunk safety function used in NSC, the gradient with respect to the initial noise is obtained by backpropagating through these numerical sampling steps and the output transformation. Each corrected noise is passed through this frozen sampler to regenerate the full action chunk.

E SENSITIVITY ANALYSIS

We conduct a sensitivity analysis on the key hyperparameters of NSC, the safe margin ϵ. Experiments are conducted on SafeLIBERO, VLA-Arena, and SafeDetour environments using $\pi_{0.5}$.

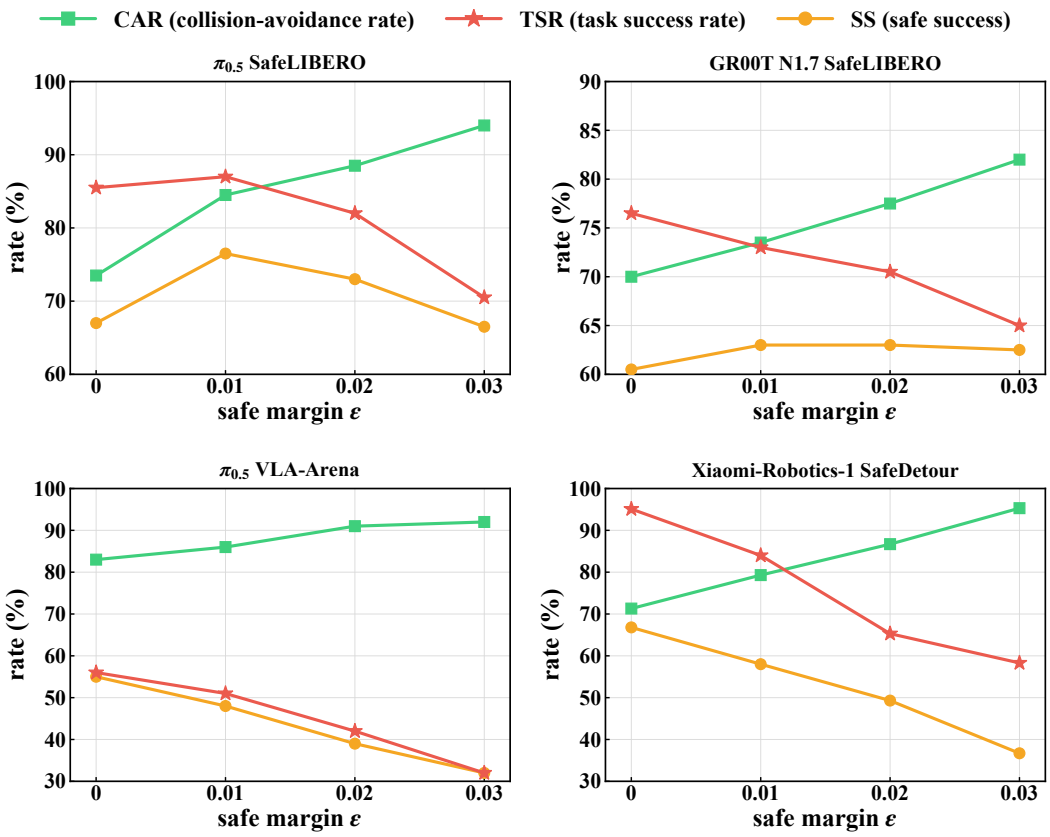


Figure 6: Sensitivity analysis for safe margin parameter ϵ

NSC certifies a chunk only when its minimum clearance satisfies $h(A) \geq \varepsilon$, where the safety margin ε is measured in meters (m) and specifies the minimum required clearance from obstacles. We sweep $\varepsilon \in \{0, 0.01, 0.02, 0.03\}$ m with all other settings fixed, on four policy and benchmark pairs: $\pi_{0.5}$ on SafeLIBERO (`object` suite, level II), GR00T N1.7 on SafeLIBERO (`spatial` suite, level II), $\pi_{0.5}$ on VLA-Arena (`static obstacles` suite), and Xiaomi-Robotics-1 on SafeDetour (`Cabinet` task). We report TSR, CAR and SS as defined in Sec. 5.1. Figure 6 shows the results.

Increasing the margin trades task success for collision avoidance on all four benchmarks. From $\varepsilon = 0$ to 0.03, CAR rises monotonically and TSR falls over the same range. the only exception is a small rise for $\pi_{0.5}$ on SafeLIBERO at $\varepsilon = 0.01$. A larger margin requires greater clearance from obstacles, reducing collision risk but also restricting the motions available for task completion.

F QUALITATIVE COMPARISON ON SAFEDETOUR

Cabinet. The robot places an object in a cabinet with glasses at the front and a wine glass at the base policy's usual placement spot. In the top row of Figure 7, NSC takes a collision-free detour and places the object at another free spot. SFM avoids collisions but continues moving without completing the placement. VLSA initially avoids the obstacles, but the manipulated object collides with them during placement.

Stove. The target pan sits on a rear burner, with a kettle and a pot staggered on the front burners. Along the trap route, avoiding only the nearer obstacle can lead the robot toward the next. NSC follows the detour route toward the rear and places the object in the pan without collisions. SFM avoids the obstacles but fails to complete the placement, whereas VLSA completes the task while colliding with the kettle.

Toaster. The base policy usually closes the toaster-oven door by lifting it at a corner. A bottle stands at one corner, and another stands behind the path to the opposite corner, so avoiding only the first bottle can lead to a collision with the second. NSC changes its approach and moves the gripper toward the middle of the door while lifting it, closing the door without collisions. SFM fails to grasp and close the door, whereas VLSA closes it but knocks over a bottle.

Interpretation. Across the illustrated rollouts, NSC generates an alternative action that enables safe task completion. These behaviors are consistent with evaluating safety over the entire predicted chunk and correcting the initial noise to regenerate the full chunk through the frozen policy. The examples illustrate how noise-space correction can address later hazards while retaining the coordinated actions needed for task completion, complementing the aggregate results in Table 3.



Figure 7: **Qualitative comparison on SafeDetour.** Rows show Cabinet, Stove, and Toaster; columns show SFM, NSC (ours), and VLSA. In these selected rollouts, SFM does not complete the task, NSC succeeds via a collision-free detour, and VLSA completes the task with obstacle collisions. Arrows indicate robot arm motion.