SafeDiffuser

Safe Planning with Diffusion Probabilistic Models

2025.05.28

Team 1
Jiwon Park & Jeongyong Yang

KAIST

Previous Paper Presentation Review

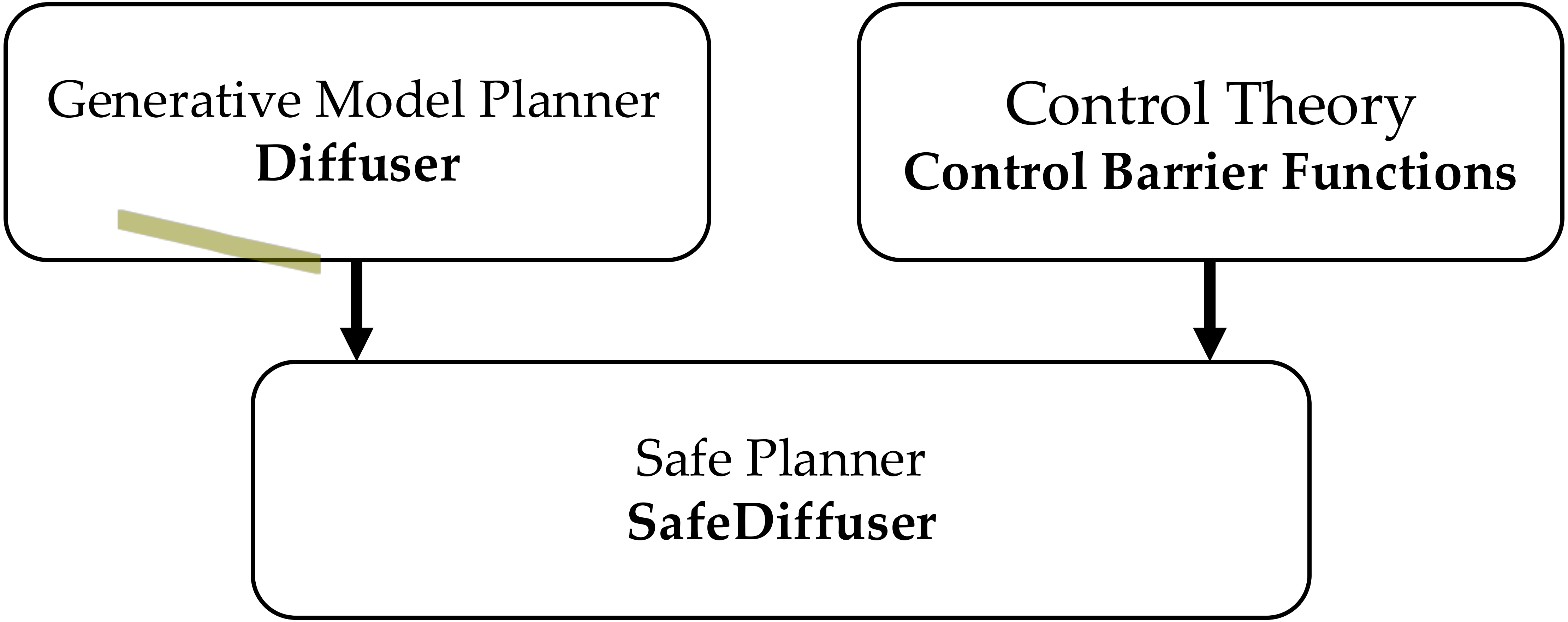
- **Paper Name: OpenVLA: An Open-Source Vision-Language-Action Model**

- Motivation
  - Existing VLA models are large (~55B parameters), closed-source, and lack fine-tuning studies
  - **OpenVLA** is 7B parameters, fully open-source, and supports efficient fine-tuning and quantization

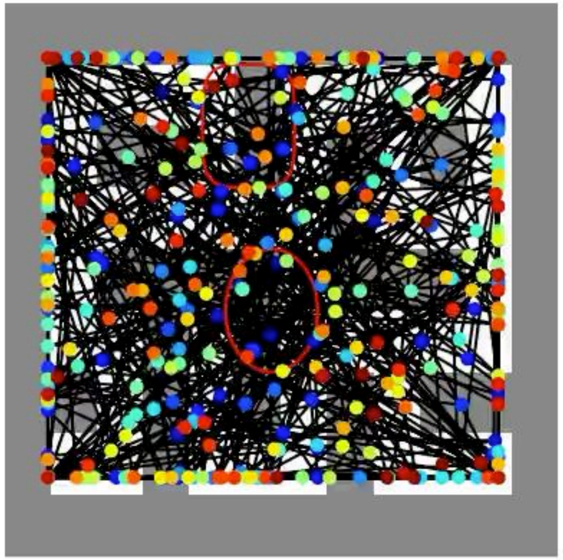
- Key Results
  - **Generalization**: Outperforms RT-2-X (55B) by 16.5% on 29 tasks with fewer parameters
  - **Fine-tuning**: Fast adaptation to new setups with just 10–150 demos
  - **Quantization**: Reduce memory with minimal performance loss

Contents

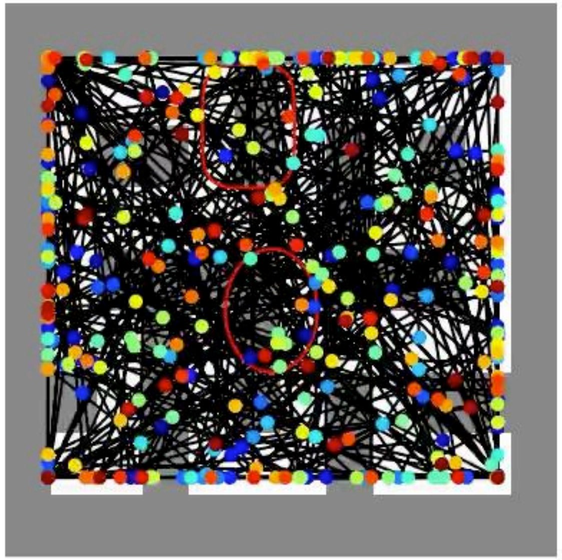
Overview of SafeDiffuser



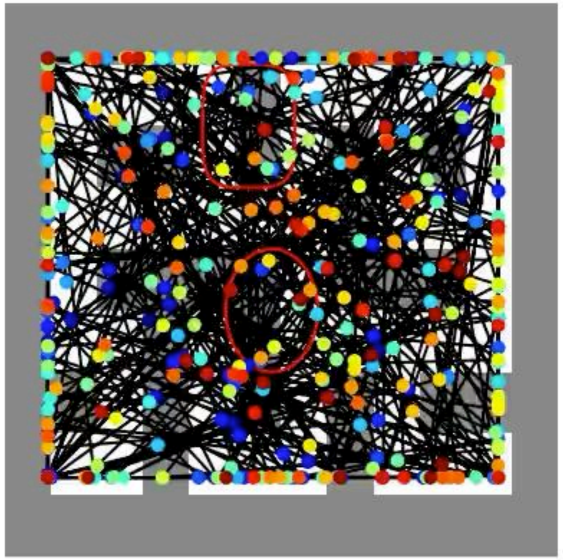
Overview of SafeDiffuser: Comparison



Diffuser

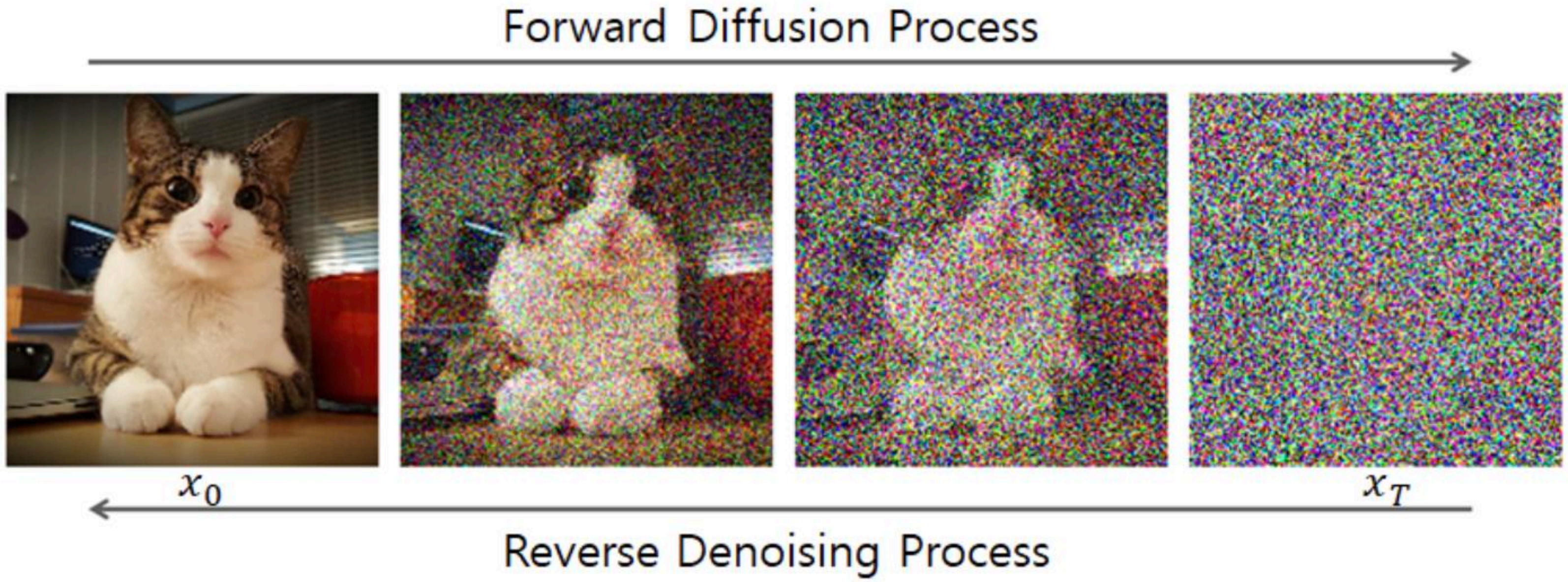


Classifier Guidance
(Potential-based)



SafeDiffuser

Generative Model Planner: Diffuser



Forward process (adding noise):

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t I)$$

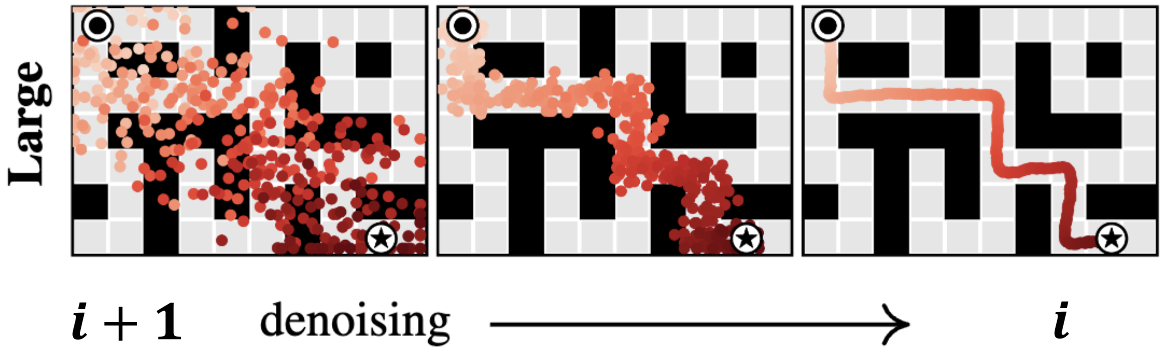
Reverse process (denoising):

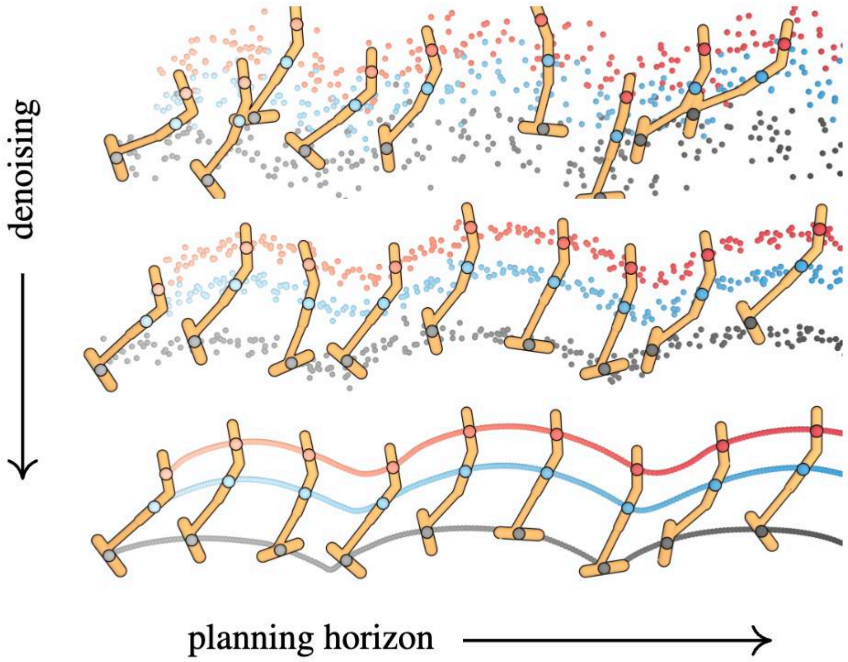
$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

KAIST

Generative Model Planner: Diffuser

- Diffuser[1]





$$\dot{x}^i = \lim_{\Delta x \to 0} \frac{x^i - x^{i+1}}{\Delta t}$$

[Diffuser Dynamics]

* i is denosing step.

Preliminaries for Control Barrier Functions

- **Def. Control affine system**:

$$\dot{x} = f(x) + g(x)u$$

where x is a state, u is a control input, and f and g are locally Lipschitz continuous function that describe the system dynamics.

- **Def. Set Invariance:**

A set $C \subset \mathbb{R}^n$ is **invariant** if:

$$x(0) \in C \rightarrow x(t) \in C, \forall t \geq 0$$

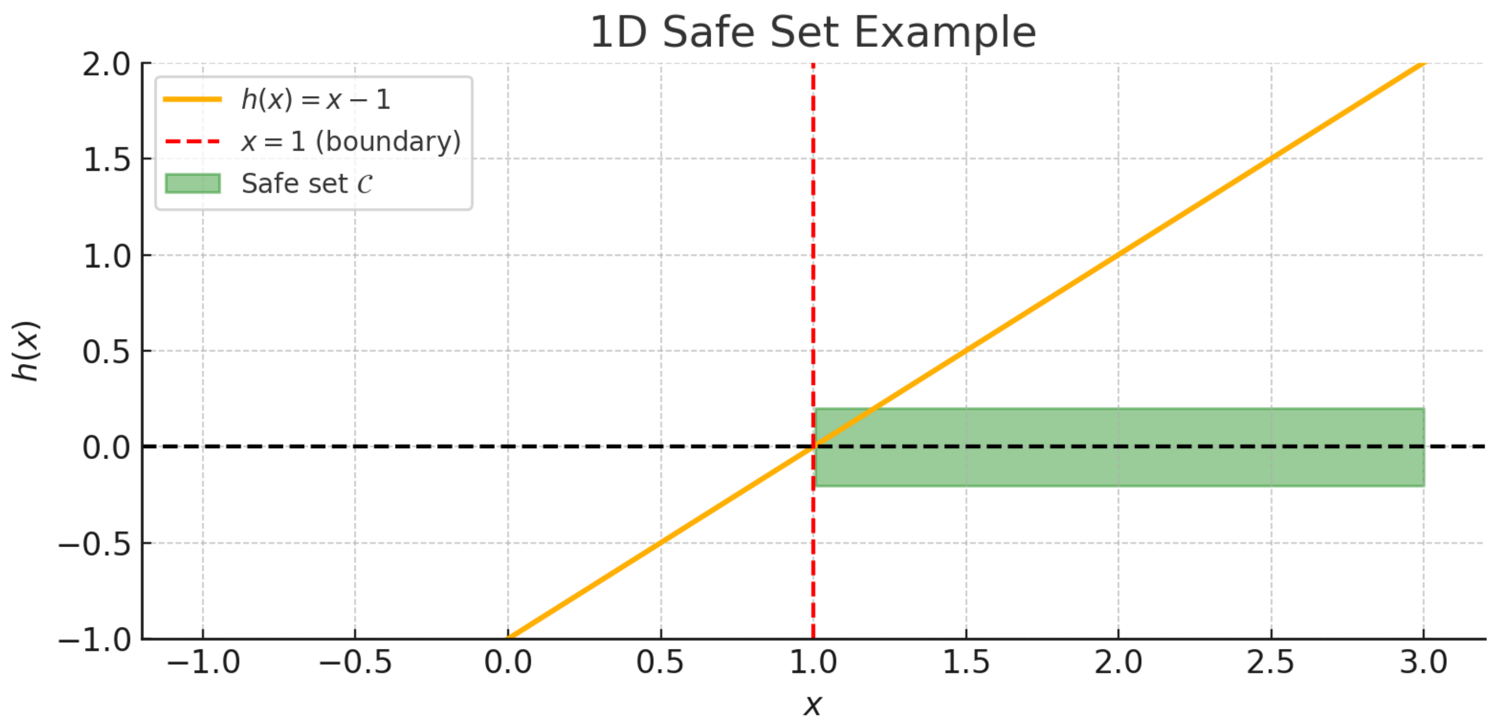
The system **remain inside** the set for all future time.

Control Barrier Functions

- **Def. Safe Set:**

Define a continuously differentiable function $h\colon \mathbb{R}^n \to \mathbb{R}$

$$C = \{x \in \mathbb{R}^n \mid h(x) \geq 0\}$$



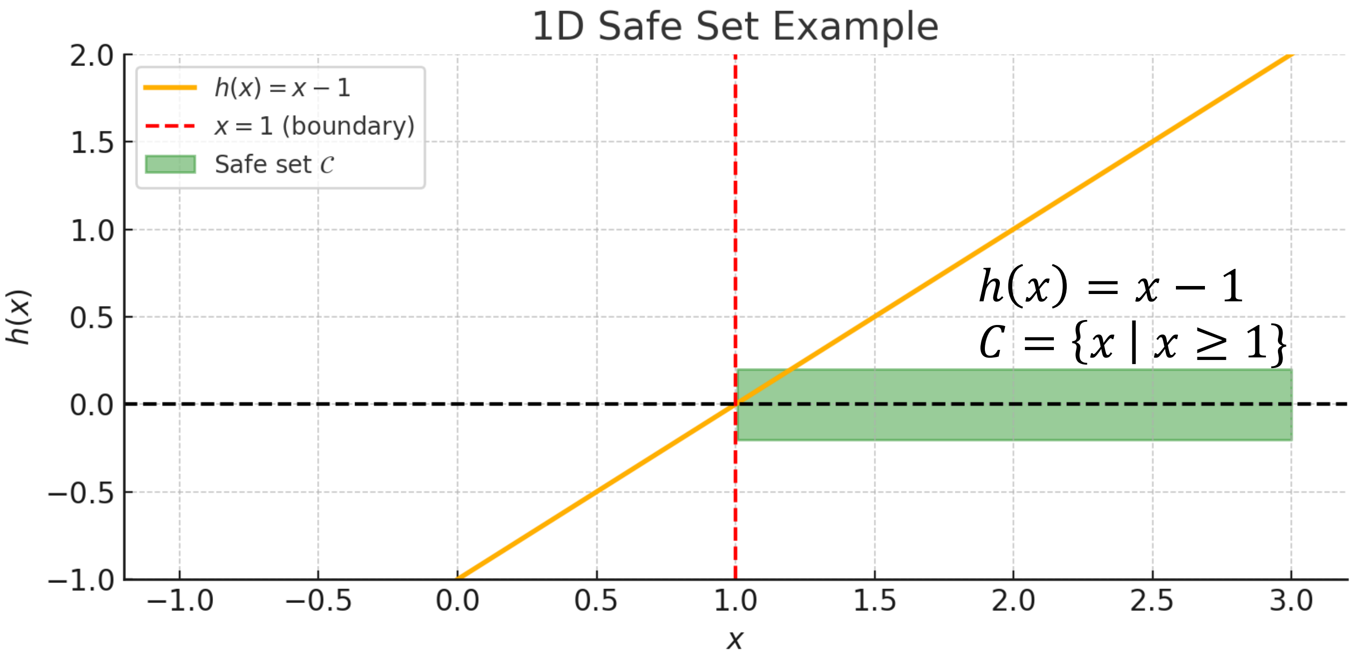
$h(x) = x - 1$
$C = \{x \mid x \geq 1\}$

Control Barrier Functions

- The idea is to make sure that, **over time, $h(x)$ doesn't drop below zero**.
- This means **the system should stay within the safe set**. To enforce this, this inequality should be satisfied:

$$\dot{h}(x) = \nabla_x h \cdot \dot{x} \geq -\alpha h(x)$$

where $\alpha > 0$.



Example (the system is $\dot{x} = u$, and choose $\alpha = 1$)

If the state is in unsafe set $C_{unsafe} = \{x | h(x) < 0\}$:

$$\dot{h}(x) = 1 \cdot u \geq -h(x)$$

If the state is at the boundary $\partial C = \{x | h(x) = 0\}$:

$$\dot{h}(x) = 1 \cdot u \geq 0$$

If the state is in safe set $C' = \{x | h(x) > 0\}$:

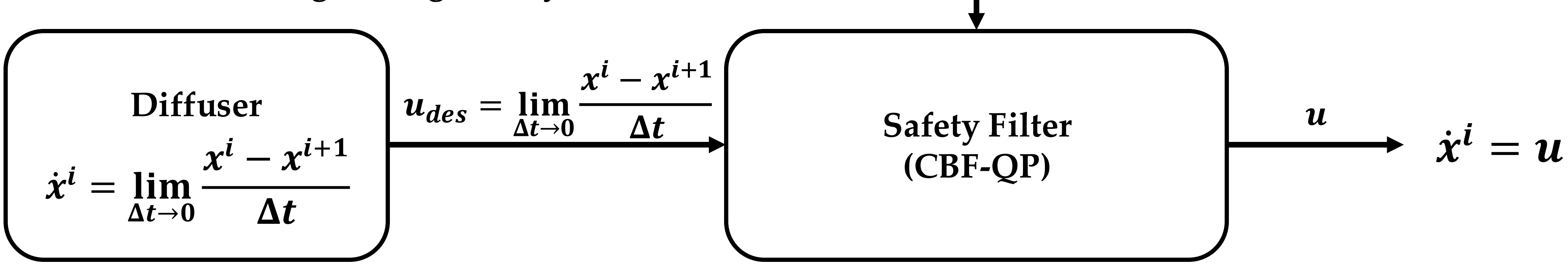
$$\dot{h}(x) = 1 \cdot u \geq -h(x)$$

Control Barrier Functions

- **Goal: Design a control input $\boldsymbol{u}$ that:**
  - Keeps the system **safe** using a Control Barrier Function.
  - Follows a desired control input u_{des} as closely as possible.
- **Optimization: CBF-QP**

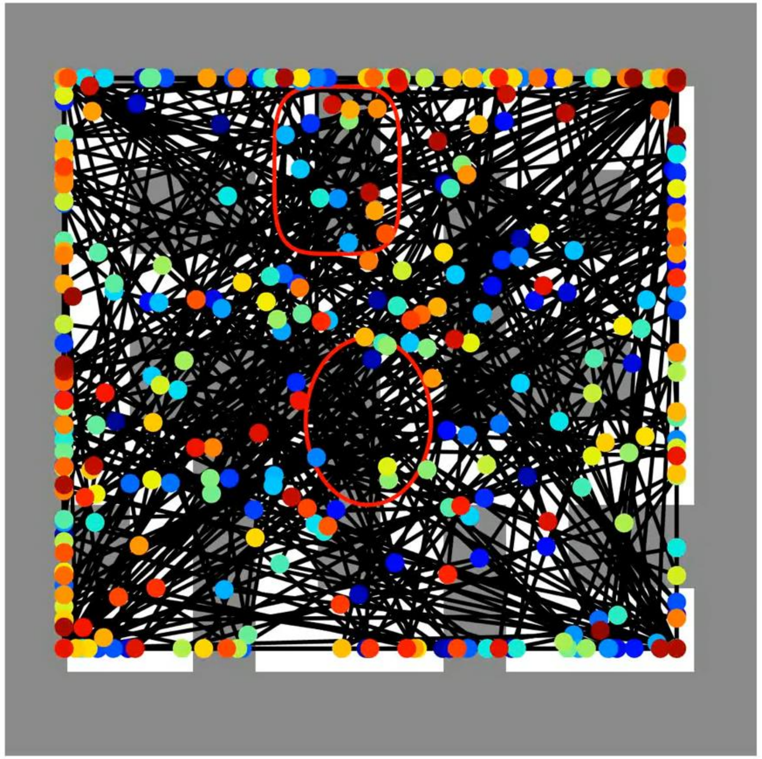
$$\min_{u} \|u - u_{des}\|^2$$
$$s.t.\ \nabla_x h(x) \cdot u + \alpha h(x) \geq 0$$

Diffuser is also single integrator system ($\dot{x} = \boldsymbol{u}$)



SafeDiffuser: Local Traps

- Local traps occur when trajectories are safe but unable to reach the goal.



SafeDiffuser: Local Traps

- They add relaxation term in the optimization problem, to allow the planner violates the safety constraint in the early phase of planning.

Constraint: $\nabla_x h(x) \cdot u + \alpha h(x) \geq 0$

↓

Constraint: $\nabla_x h(x) \cdot u + \alpha h(x) \geq -\delta(i)$

$where\ \delta(i) \geq 0$

SafeDiffuser: Local Traps

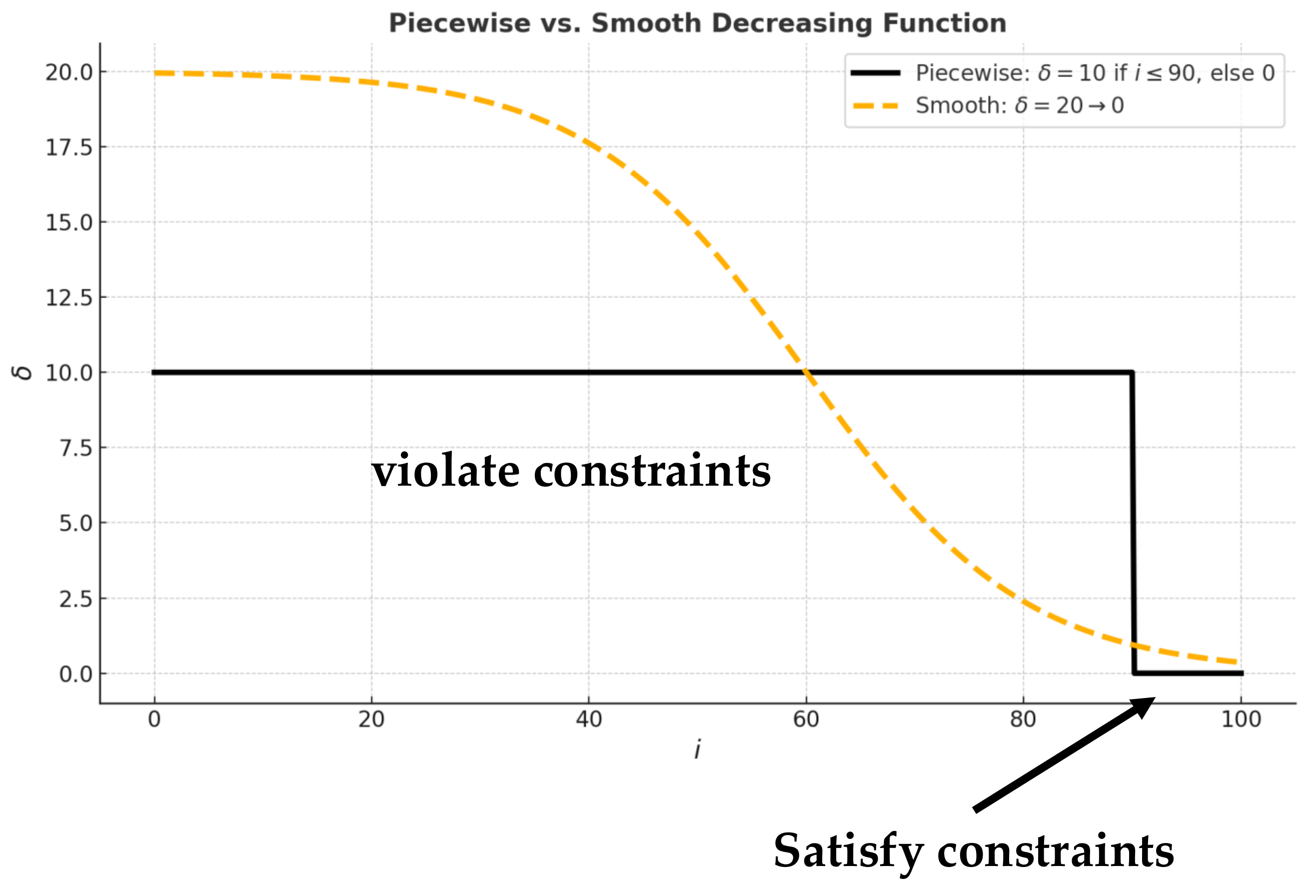
- Relaxed and time-varying SafeDiffuser help the planner escape local traps.

$$\min_{u,r} \|u - u_{des}\|^2 + \|r\|^2$$
$$s.t.\ \nabla_x h(x) \cdot u + \alpha h(x) \geq -w(i) r$$

ReS
(Relaxed SafeDiffuser):

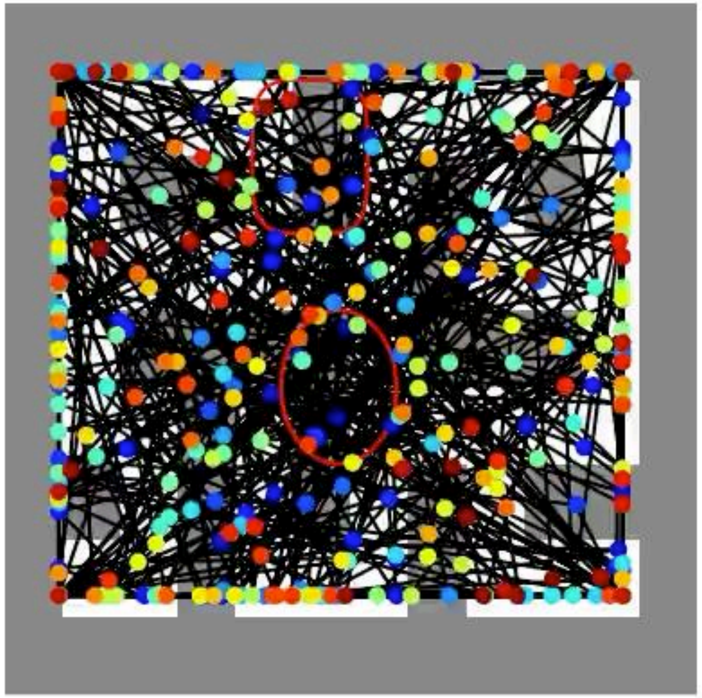
$$\min_{u} \|u - u_{des}\|^2$$
$$s.t.\ \nabla_x h(x) \cdot u + \alpha h(x) \geq -\alpha\gamma(i) - \dot{\gamma}(i)$$

TVS
(Time-Varying SafeDiffuser):

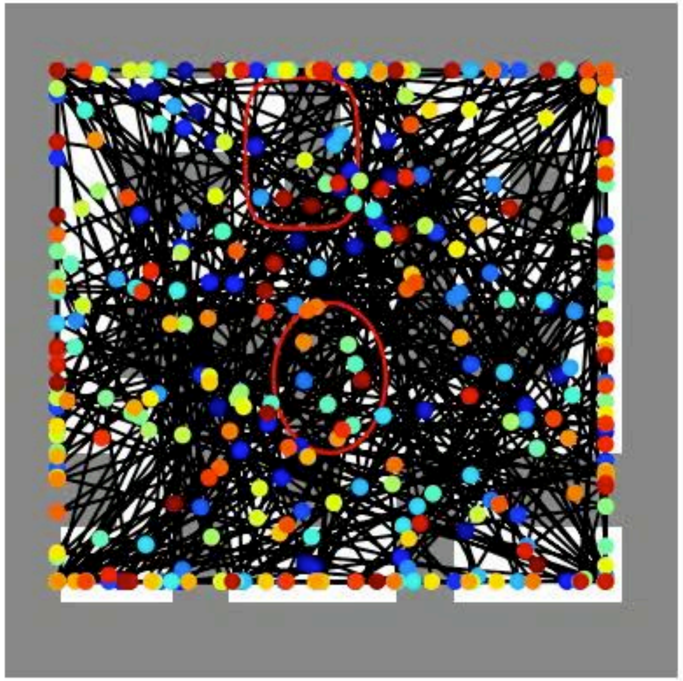


Experiment Results: Maze2D

- Diffuser cannot generate safe path.
- Basic SafeDiffuser can avoid safety constraints, but local traps occur.



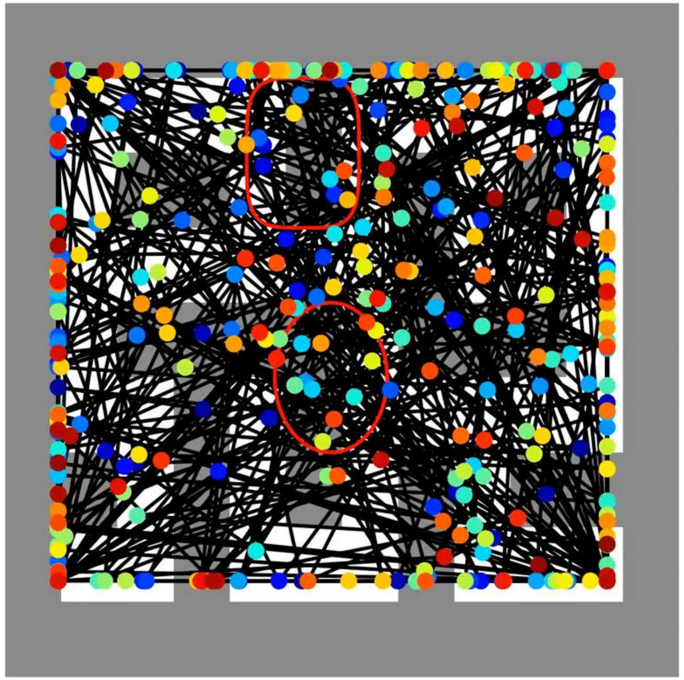
Diffuser



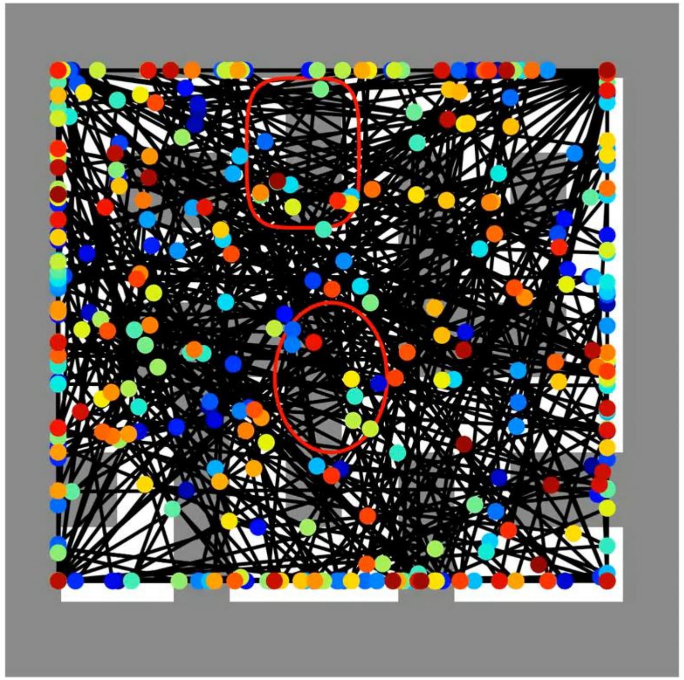
SafeDiffuser: RoS
(Basic version - Local trap occurs)

Experiment Results: Maze2D

- Relaxed SafeDiffuser and Time-varying SafeDiffuser can resolve local trap problems.



ReS



TVS

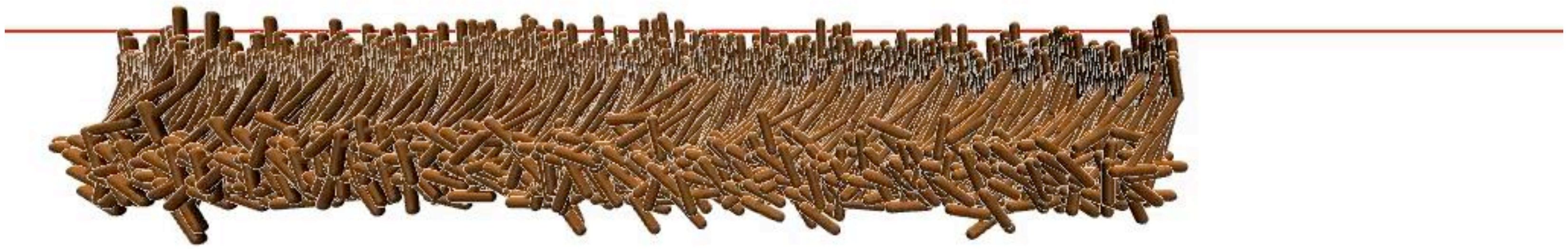
KAIST

Experiment Results: Maze2D

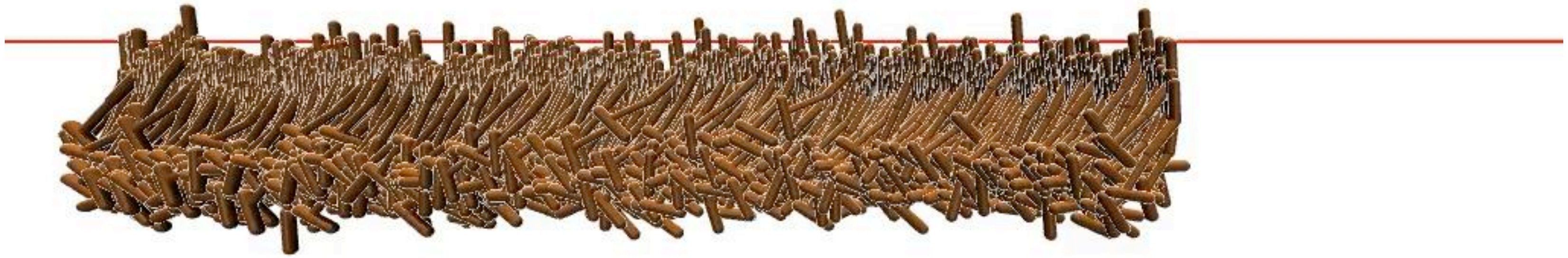
Method	S-spec(↑ & ≥ 0)	C-spec(↑ & ≥ 0)	Score (↑)	Time	NLL	Trap rate 1 (↓)	Trap rate 2 (↓)
Diffuser Janner et al. (2022)	-0.983	-0.894	1.598±0.174	0.006	4.501±0.475		
Trunc. Brockman et al. (2016)	$-1.192e^{-7}$	-0.759	1.577±0.242	0.024	4.494±0.465		
CG Dhariwal & Nichol (2021)	-0.789	-0.979	0.384±0.020	0.053	6.962±0.350		
CG−ε Dhariwal & Nichol (2021)	-0.853	-0.995	0.383±0.017	0.061	6.975±0.343		
InvODE Xiao et al. (2023b)	14.000	$1.657e^{-5}$	-0.025±0.000	0.018	–		
RoS-diffuser (Ours)	0.010	0.010	1.519±0.330	0.106	4.584±0.646	100%	100%
RoS-diffuser-cf (Ours)	0.010	0.010	1.536±0.306	0.007	4.481±0.298	100%	100%
ReS-diffuser (Ours)	0.010	0.010	1.557±0.289	0.107	4.434±0.561	46%	17%
ReS-diffuser-cf (Ours)	0.010	0.010	1.544±0.280	0.007	4.619±0.652	36%	16%
TVS-diffuser (Ours)	0.003	0.003	1.543±0.303	0.107	4.533±0.494	47%	21%
TVS-diffuser-cf (Ours)	0.003	0.003	1.588±0.231	0.007	4.462±0.431	48%	18%
ReS-diffuser-L10 (Ours)	0.010	0.010	1.527±0.291	0.011	4.571±0.693	39%	8%

[Results of SafeDiffuser]

Experiment Results: Hopper



Hopper

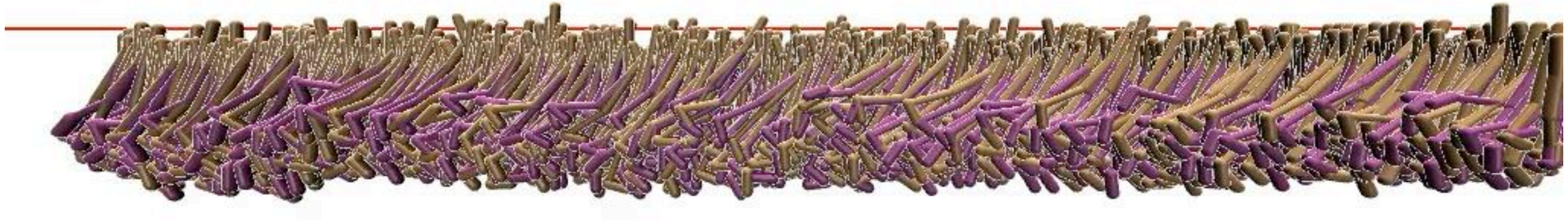


Safe Hopper

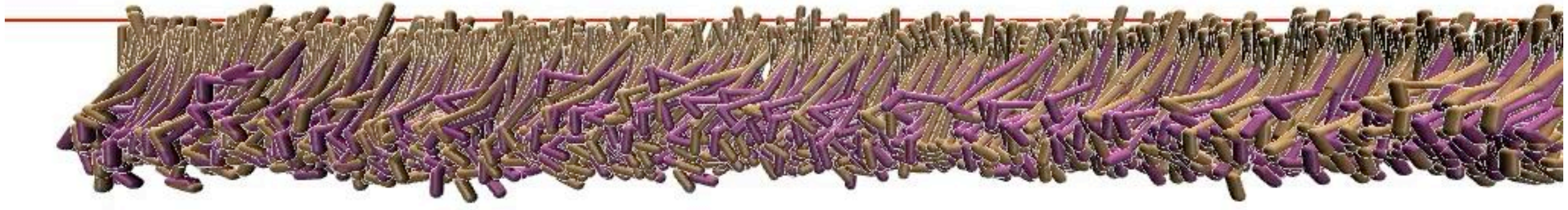
[Inference Result of SafeDiffuser[2] in Mujoco]

KAIST

Experiment Results: Walker2D



Walker2D



Safe Walker2D

[Inference Result of SafeDiffuser[2] in Mujoco]

KAIST

Experiment Results: Hopper and Walker2D

Experiment	Method	S-spec(↑ & ≥ 0)	C-spec(↑ & ≥ 0)	Score (↑)	Time
Walker2D	Diffuser Janner et al. (2022)	-9.375	-4.891	0.346±0.106	0.037
	Trunc. Brockman et al. (2016)	0.0	×	0.286±0.180	0.105
	CG Dhariwal & Nichol (2021)	-0.575	-0.326	0.208±0.140	0.053
	RoS-diffuser (Ours)	0.000	0.010	0.312±0.165	0.183
	RoS-diffuser-CF (Ours)	0.000	0.010	0.321±0.119	0.040
Hopper	Diffuser Janner et al. (2022)	-2.180	-1.862	0.455±0.038	0.038
	Trunc. Brockman et al. (2016)	0.0	×	0.436±0.067	0.046
	CGDhariwal & Nichol (2021)	-0.894	-0.524	0.478±0.038	0.047
	RoS-diffuser (Ours)	0.000	0.010	0.430±0.040	0.170
	RoS-diffuser-CF (Ours)	0.000	0.010	0.464±0.028	0.040

[Results of SafeDiffuser]

KAIST

References

1. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In International Conference on Machine Learning, pages 9902–9915. PMLR, 2022.
2. Wei, W. Tsun-Hsuan, G. Chuang, H. Ramin, L. Mathias, and R. Daniela. Safediffuser: Safe planning with diffusion probabilistic models. IEEE, 2025.
3. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. arXiv preprint arXiv:2210.02747, 2022.
4. P. Bhat and D. S. Bernstein. Finite-time stability of continuous autonomous systems. SIAM Journal on Control and optimization, 38(3):751–766, 2000.

Thank you