NEURIPS 2026 · SAND

Skill-Adaptive Noise Scheduling for Diffusion Policies

Woo Kyung Kim · Gwangpyo Yoo · Eunyoung Park · Honguk Woo

Sungkyunkwan University

PROBLEM

One fixed schedule for every skill and every dimension

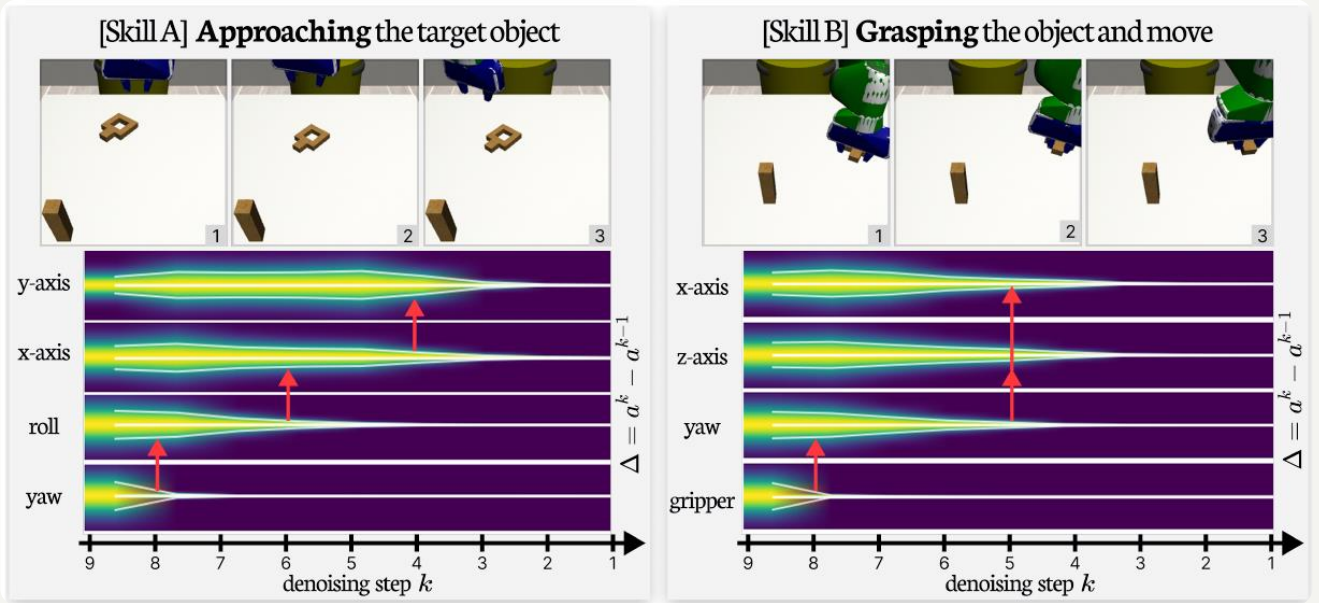
Standard forward process

$$q(x^k \mid x) = \mathcal{N}(x^k;\ \alpha_k\, x, [\sigma(k)]^2\, I)$$

One scalar pair (α_k, σ_k) per step, shared by all N action dimensions and every skill.

Why it falls short

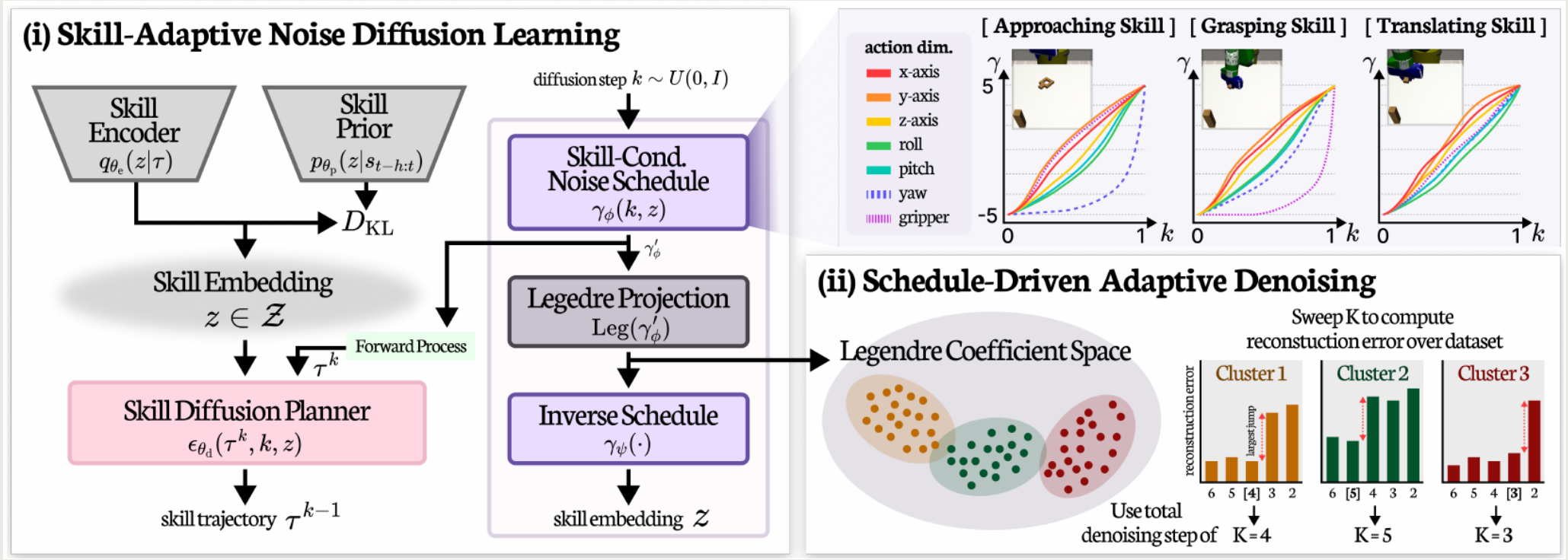
- Loose bound on the diffusion ELBO
- Deterministic sampling: early errors persist, and more steps do not undo them



What SaND learns instead: approaching settles yaw and roll first, grasping commits the gripper first. Position is refined later.

METHOD · OVERALL FRAMEWORK

SaND Framework



(i) Skill-Adaptive Noise Diffusion learns a skill-conditioned noise schedule with an inverse schedule. **(ii) Schedule-Driven Adaptive Denoising** sets K per skill from the schedule's shape.

METHOD

Skill-Adaptive Noise Diffusion

SCHEDULE

Skill embedding
z from skill encoder

→

Schedule network
$g_\varphi\colon z \to C$

→

Noise schedule
$\gamma_\varphi(k, z)$ per action dim

→

Diffusion planner
per-dim α_k, σ_k

INVERSE

Schedule derivative
$\gamma'_\varphi(k, z)$

→

Legendre projection
$Leg(\gamma'_\varphi)$

→

Inverse schedule
$\gamma_\psi(\cdot)$

→

Recovered skill
$\hat{z} \approx z$ (prototype)

Noise Schedule

$$g_\varphi\colon \mathbb{R}^{d_z} \to \mathbb{R}^{6\times N}, f(k, C) = \sum_{n=0}^{5} C_n\, k^n, \gamma_\varphi(k, z) = f(k;\, g_\varphi(z)) \in \mathbb{R}^N$$

Monotone 5th-degree polynomial per action dimension, with $\gamma_\varphi(0, z) = \gamma_{\min}$ and $\gamma_\varphi(1, z) = \gamma_{\max}$.

Diffusion Loss

$$L_{du}(\theta_e, \theta_d, \varphi) = E_{k\sim U[0,1], \varepsilon\sim\mathcal{N}(0,I)} \left[\gamma'_\varphi(k, z) \left\| \varepsilon_{\theta_d}\left(\tau^k, k, sg(z)\right) - \varepsilon \right\|_2^2\right] + \beta\, D_{KL}(q_{\theta_e}(z|\tau), p(z))$$

$\gamma'_\varphi = \partial_k \gamma_\varphi$ reweights the DSM loss; $sg(z)$ blocks the direct gradient path to the skill encoder.

Inverse Schedule

$$L_{inv}(\varphi, \psi) = E \left\| \gamma_\psi\left(\mathrm{Leg}\left(\gamma'_{\varphi(k,z)}\right)\right) - z \right\|_2^2$$

Recovering z from the Legendre coefficients keeps the schedule from degenerating into a fixed one.

METHOD

Schedule-Driven Adaptive Denoising

OFFLINE

Every skill in the data
$\mathrm{Leg}(\gamma'(\cdot, z))$

→

K-means
cluster schedule shapes

→

Sweep K downward
reconstruction MSE

→

Kneedle
K before the MSE jump

ONLINE

Skill prior
z from past h states

→

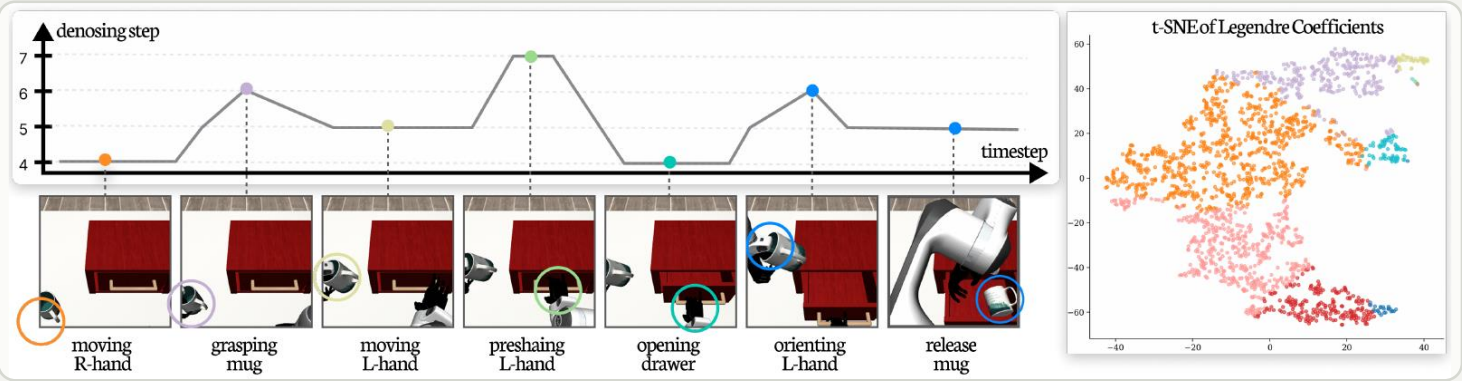
Legendre code
$\mathrm{Leg}(\gamma'(\cdot, z))$

→

Nearest centroid
cluster c

→

Denoise
with that cluster's K



Drawer task: reaching and opening use **4 steps**; grasping, preshaping and orienting use **6–7**.

The lookup is negligible: the learned schedule adds 0.31% FLOPs.

EXPERIMENT RESULTS

Higher success on contact-rich and dexterous tasks

Average success rate (%)

6 tasks per benchmark · 5 seeds · 95% CI · baselines use a fixed cosine schedule

Method	K	MimicGen	DexMimicGen
DP	10	51.4 ± 1.4	30.3 ± 3.9
DV	10	52.5 ± 3.6	46.0 ± 5.6
SkillDiffuser	10	52.3 ± 1.9	51.1 ± 4.3
LDuS	10	49.6 ± 0.5	47.1 ± 3.4
SaND	10	64.0 ± 1.0	64.1 ± 2.7
SaND+AD	3.7 / 5.4	63.3 ± 0.8	63.8 ± 1.9

Bi-DexHands (52-D actions, reward, K = 5): SaND 349.3 vs DV 313.3

+11.5 / +13.0 pp

success over the best baseline on MimicGen (DV) / DexMimicGen (SkillDiffuser)

10 → 3.7 / 5.4

avg. denoising steps with adaptive denoising (−63.2% / −45.6%), success −0.7 / −0.3 pp

5.07 → 2.60 ms

per skill trajectory on an RTX 4090; FLOPs 777 M → 335 M (−56.8%)

EXPERIMENT RESULTS

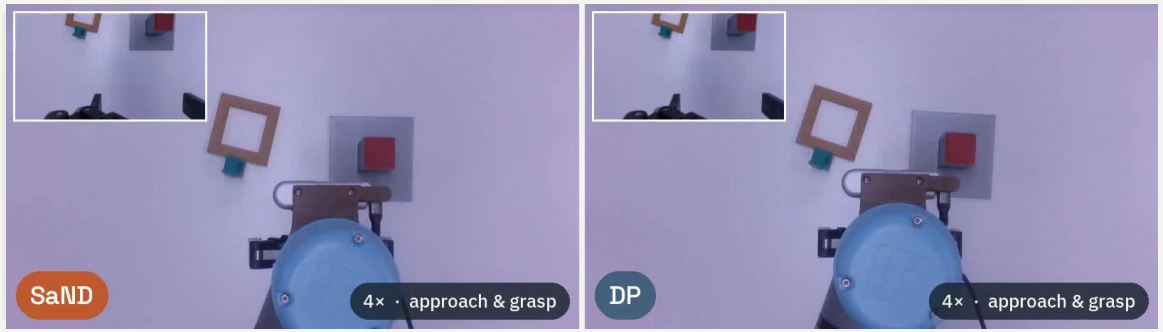
Ablation Study and Real-World Experiment

Ablation · avg. success (%)

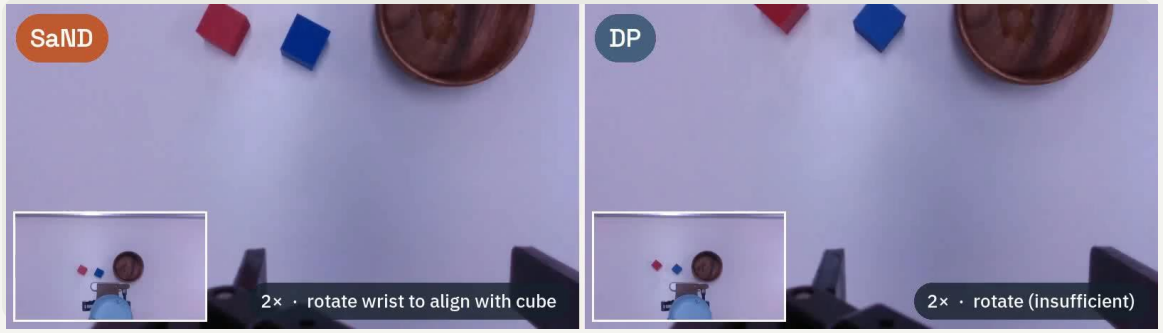
Variant	MimicGen	DexMimicGen
SaND (full)	64.0	64.1
− Legendre projection	61.3 (−2.7)	60.9 (−3.2)
− Inverse schedule	60.4 (−3.6)	60.7 (−3.4)
− Learned schedule	57.6 (−6.4)	57.0 (−7.1)

Adaptive denoising · similar step budget

Steps chosen by	MimicGen	DexMimicGen
Legendre clusters	63.3	63.8
Skill-embedding clusters	61.7 (−1.6)	61.2 (−2.6)
Fixed K	61.3 (−2.0)	61.2 (−2.6)



Nut insertion: DP 20% → SaND 90% (SaND+AD 80%)



Cube stacking: DP 20% → SaND 60% (SaND+AD 70%)

TAKEAWAYS

Three things to remember

1 Learn the noise schedule per skill and per action dimension.

+11.5 pp on MimicGen and +13.0 pp on DexMimicGen over the strongest baseline

2 Keep it skill-specific with an inverse schedule on its Legendre coefficients.

Without it, the schedule drifts back toward one fixed curve

3 Reuse the schedule’s shape to set each skill’s number of denoising steps.

46–63% fewer steps, with under 1 pp loss in success

Limitation: the schedule needs a skill embedding to condition on, so non-skill diffusion policies would need an equivalent context signal.