Enhancing Control Policy Smoothness by Aligning Actions with Predictions from Preceding States: Appendix

A Theoretical Results

A.1 Proof: Loss for Composite Lipschitz Constraint

Theorem 1 (Composite Lipschitz Constraint)**.** *Under Assumptions 1-2, for all* $(s_{t-1}, a_{t-1}) \in \mathcal{S} \times \mathcal{A}$ *and* $\xi_{t-1}^{(1)}, \xi_{t-1}^{(2)} \in \Xi$*, it holds that*

$$d_A(f \circ T(s_{t-1}, a_{t-1}, \xi_{t-1}^{(1)}),\ f \circ T(s_{t-1}, a_{t-1}, \xi_{t-1}^{(2)})) \\ \leq K\, d_S(T(s_{t-1}, a_{t-1}, \xi_{t-1}^{(1)}),\ T(s_{t-1}, a_{t-1}, \xi_{t-1}^{(2)})), \\ \forall\, \xi_1, \xi_2 \in \Xi. \tag{A.1-1}$$

Therefore, the composite function $f \circ T$ *is locally Lipschitz continuous in the noise* ξ*, with local Lipschitz constant*

$$K_{\text{comp}}(s_{t-1}, a_{t-1}) \ \leq\ K\, K_\xi(s_{t-1}, a_{t-1}). \tag{A.1-2}$$

Proof. By the Lipschitz continuity of f in the state argument, we have

$$d_A(f(T(s_{t-1}, a_{t-1}, \xi^{(1)})), f(T(s_{t-1}, a_{t-1}, \xi^{(2)}))) \\ \leq K\, d_S(T(s_{t-1}, a_{t-1}, \xi^{(1)}), T(s_{t-1}, a_{t-1}, \xi^{(2)})). \tag{A.1-3}$$

Applying Assumption 1 to the right-hand side gives

$$d_S(T(s_{t-1}, a_{t-1}, \xi^{(1)}), T(s_{t-1}, a_{t-1}, \xi^{(2)})) \\ \leq K_\xi(s_{t-1}, a_{t-1})\, d_\Xi(\xi^{(1)}, \xi^{(2)}), \tag{A.1-4}$$

and chaining the two inequalities yields

$$d_A(f \circ T(s_{t-1}, a_{t-1}, \xi^{(1)}), f \circ T(s_{t-1}, a_{t-1}, \xi^{(2)})) \\ \leq K\, K_\xi(s_{t-1}, a_{t-1})\, d_\Xi(\xi^{(1)}, \xi^{(2)}). \tag{A.1-5}$$

This establishes the claimed local Lipschitz property with constant $K_{\text{comp}}(s_{t-1}, a_{t-1}) = K\, K_\xi(s_{t-1}, a_{t-1})$. □

A.2 Proof: Loss for Composite Lipschitz Constraint

This paper defines the per-state smoothness loss as

$$L \ =\ \left\|\pi_\phi(s_t) - \mathbb{E}_{\tilde{s}_t \sim P(\cdot|s_{t-1})}\left[\pi_\phi(\tilde{s}_t)\right]\right\|_2^2. \tag{A.2-1}$$

Intuitively, L penalises the deviation of the current action $\pi_\phi(s_t)$ from the *mean action* taken over all next-state samples $\tilde{s}_t$ that originate from the same (s_{t-1}, a_{t-1}).

Proof. Throughout this section we equip the action space $\mathcal{A} \subset \mathbb{R}^m$ with the Euclidean metric $d_A(a, a') := \|a - a'\|_2$. Let two independent noise samples $\xi_{t-1}^{(1)}, \xi_{t-1}^{(2)}$ generate

$$s_t := T\big(s_{t-1}, a_{t-1}, \xi_{t-1}^{(1)}\big), \\ \tilde{s}_t := T\big(s_{t-1}, a_{t-1}, \xi_{t-1}^{(2)}\big), \tag{A.2-2}$$

and denote $\mu_t = \mathbb{E}_{\tilde{s}_t \sim P(\cdot|s_{t-1})}\big[\pi_\phi(\tilde{s}_t)\big]$. For the fixed pair $(\xi_{t-1}^{(1)}, \xi_{t-1}^{(2)})$ the triangle inequality gives

$$\|\pi_\phi(s_t) - \pi_\phi(\tilde{s}_t)\|_2 \\ \leq\ \|\pi_\phi(s_t) - u_t\|_2 + \|u_t - \pi_\phi(\tilde{s}_t)\|_2 \\ =\ \sqrt{L} + C, \quad C := \|u_t - \pi_\phi(\tilde{s}_t)\|_2. \tag{A.2-3}$$

Rewriting the classical definition of Lipschitz constant of the composite function with respect to noise for Eq. A.2-3 yields the following:

$$K_{\text{comp}} := \sup_{\xi_{t-1}^{(1)} \neq \xi_{t-1}^{(2)}} \frac{\|\pi_\phi(s_t) - \pi_\phi(\tilde{s}_t)\|_2}{d_\Xi(\xi_{t-1}^{(1)}, \xi_{t-1}^{(2)})} \\ \leq \sup_{\xi_{t-1}^{(1)} \neq \xi_{t-1}^{(2)}} \frac{\sqrt{L} + C}{d_\Xi(\xi_{t-1}^{(1)}, \xi_{t-1}^{(2)})}. \tag{A.2-4}$$

Because the noise distribution P_Ξ is fixed during training, the denominators in Eq. A.2-4 are constant. Hence *minimising the loss* L directly lowers the upper bound on K_{comp}.

Next, Theorem 1 and Jensen's inequality bound the residual term by

$$C = \|u_t - \pi_\phi(\tilde{s}_t)\|_2 \ \leq\ 2\sigma_\xi\, K_{\text{comp}}, \tag{A.2-5}$$

so the same training step that reduces K_{comp} also shrinks C. Combining Eq. A.2-4 and Eq. A.2-5 shows that both the loss Eq. A.2-1 and its induced residual decrease monotonically drive down the composite Lipschitz constant. □

B Implementation Details

B.1 Hardware / Software Information

Component	Configuration
Gymnasium	1.0.0
Stable-Baselines3	2.5.0
mujoco	3.3.3
PPO training GPU	NVIDIA RTX 3090
SAC training GPU	NVIDIA A5000 and A4000 (mixed)

Table B.1: Hardware and software used for Gymnasium based task.

Gymnasium. See Table B.1 for the hardware and software configurations used in Gymnasium.

Component	Configuration
Isaac-Lab version	2.1.0
Isaac-Sim version	4.5.0
Execution GPU	NVIDIA A4000

Table B.2: Hardware and software details for Isaac-Lab based task.

Isaac-Lab. See Table B.2 for the hardward and softward configurations used in Isaac-lab.

B.2 Effectiveness on Transition-induced Similar State

To demonstrate the effect of transition-induced similar states, we ran the following sequence of experiments in Gymnasium environments with a SAC setup:

1. train a policy to convergence using SAC.
2. collect a large replay buffer dataset with the trained policy.
3. train a ASAP predictor via supervised learning on the fixed buffer data.
4. fine-tune the RL algorithm by incorporating the predictor's outputs into the policy loss.

The total training and fine-tuning time steps are summarized in Table B.3. Unless otherwise noted, all other settings follow the Gymnasium benchmark (see Appendix B.3).

	Train Timestep	Fine-Tune Timestep
Lunar Lander	500k	200k
Reacher	500k	200k
Hopper	1M	200k
Walker	1M	100k

Table B.3: Training and fine-tuning time steps.

B.3 Evaluation on Gymnasium

All code for the Gymnasium framework (Towers et al. 2024) was implemented using Stable Baselines3 (SB3) (Raffin et al. 2021). We evaluated six benchmark tasks: LunarLander-v3, Pendulum-v1, Reacher-v5, Ant-v5, Hopper-v5, and Walker-v5, using PPO (Schulman et al. 2017) and SAC (Haarnoja et al. 2018). For SAC, we applied the CAPS (Mysore et al. 2021) settings directly, whereas for PPO we used SB3 defaults due to implementation incompatibilities. Detailed hyperparameters for PPO and SAC can be found in Table B.4, Table B.5 and Table B.6. The baselines included vanilla SAC/PPO, CAPS, L2C2 (Kobayashi 2022), and Grad-CAPS (Lee et al. 2024). CAPS was ported to SB3 from its public codebase, while L2C2 and Grad-CAPS were reimplemented based on their original papers due to lack of released code.

For the Gymnasium experiments, we applied each method's hyperparameters exactly as described in the original papers when available. If no environment-specific settings were provided, we followed the hyperparameter selection procedures outlined in those papers. Training used the same 10 random seeds across all tests and evaluation used the same 100 random seeds. The exact random seeds are included with the accompanying code.

Parameter	Default Value
learning rate	3e-4
n steps	2048
batch size	64
n epochs	10
gamma	0.99
gae lambda	0.95
clip range	0.2
clip range vf	None
normalize advantage	True
ent coef	0.0
vf coef	0.5
max grad norm	0.5
use sde	False
target kl	None
hidden layer in actor network	[64, 64]
hidden layer in critic network	[64, 64]
activation	Tanh
optimizer	Adam(eps:1e-5)

Table B.4: Parameters for PPO setting

Parameter	Default Value
n envs	1
learning rate	1e-3
buffer size	1e6
learning starts	10000
batch size	100
tau	0.005
gamma	0.99
train freq	50
gradient steps	50
action noise	None
ent coef	0.2
target update interval	1
target entropy	auto
optimize memory usage	False
use sde	False
hidden layer in actor network	[256, 256]
hidden layer in critic network	[256, 256]
activation	ReLU
optimizer	Adam(eps:1e-8)

Table B.5: Parameters for SAC setting

	PPO	SAC
Lunar Lander	500k	500k
Pendulum	500k	100k
Reacher	500k	500k
Ant	1M	1M
Hopper	1M	1M
Walker	1M	1M

Table B.6: Training time steps for PPO and SAC setting.

Hyper Parameter per Environment. This section summarises the selection strategy and final settings used for each method in every environment.

- CAPS
  - *Pendulum, LunarLander, Reacher, Ant*: the SAC and PPO parameters supplied in the original CAPS paper were adopted without modification.
  - *Hopper, Walker*: because they share the same MuJoCo physics engine, we reused the parameters reported for *Reacher* and *Ant*.
- L2C2
  - For both SAC and PPO we followed the original L2C2 procedure: $\overline{\lambda}$ and $\underline{\lambda}$ were derived from CAPS's λ_T, and the noise standard deviation σ was set to the value recommended by the authors.
- Grad-CAPS
  - PPO: retained CAPS's λ_T exactly as in the Grad-CAPS paper.
  - SAC: fixed $\lambda_T = 1$ across all environments as in the Grad-CAPS paper.
- ASAP
  - PPO: manually tuned hyper-parameters; soft update for predictor was not used.
  - SAC: Manually tuned hyper-parameters; soft update for predictor was used with $\tau = 0.01$ and the target-update interval fixed to 1.

Refer to Tables B.7 and B.8 for the detailed hyper parameters.

B.4 Applicability in Robotics Scenarios

All experiments were implemented with the rl-games (Makoviichuk and Makoviychuk 2021) library on top of the Isaac-Lab (Mittal et al. 2023) framework. We evaluated our approach on four benchmark tasks: Isaac-Reach-Franka-v0, Isaac-Lift-Cube-Franka-v0, Isaac-Repose-Cube-Allegro-v0, and Isaac-Velocity-Rough-Anymal-C-v0. The base policy was PPO, and results are therefore reported only in comparison with the original PPO baseline. All evaluations used the default Isaac-Lab training environments, which already incorporate domain randomization and observation noise. The detailed PPO hyper-parameters strictly follow the default values provided by Isaac-Lab; for the full architecture and environment settings, see https://github.com/isaac-sim/IsaacLab/tree/v2.1.0. The observation noise values can be found in Tables B.9, B.11, and B.12, and the detailed ASAP hyperparameters are listed in Table B.13.

Parameter	Noise Type	Range
joint_pos	Uniform	$[-0.01,\ 0.01]$
joint_vel	Uniform	$[-0.01,\ 0.01]$

Table B.9: Observation noise configuration used in the Reach-Franka task.

Parameter	Noise Type	Range
joint_pos	Gaussian	$\mu = 0.0,\ \sigma = 0.005$
joint_vel	Gaussian	$\mu = 0.0,\ \sigma = 0.01$
object_pos	Gaussian	$\mu = 0.0,\ \sigma = 0.002$
object_lin_vel	Gaussian	$\mu = 0.0,\ \sigma = 0.002$
object_ang_vel	Gaussian	$\mu = 0.0,\ \sigma = 0.002$

Table B.10: Observation noise configuration used in the Lift-Cube-Franka task.

Parameter	Noise Type	Range
joint_vel	Gaussian	$\mu = 0.0,\ \sigma = 0.01$
object_pos	Gaussian	$\mu = 0.0,\ \sigma = 0.002$
object_lin_vel	Gaussian	$\mu = 0.0,\ \sigma = 0.002$
object_ang_vel	Gaussian	$\mu = 0.0,\ \sigma = 0.002$

Table B.11: Observation noise configuration used in the Repose-Cube-Allegro task.

Parameter	Noise Type	Range
base_lin_vel	Uniform	$[-0.1,\ 0.1]$
base_ang_vel	Uniform	$[-0.2,\ 0.2]$
projected_gravity	Uniform	$[-0.05,\ 0.05]$
joint_pos	Uniform	$[-0.01,\ 0.01]$
joint_vel	Uniform	$[-1.5,\ 1.5]$
height_scan	Uniform	$[-0.1,\ 0.1]$

Table B.12: Observation noise configuration used in the Velocity-Rough-Anymal task.

B.5 Compatibility with Architectural Method

The detailed configuration follows that of Appendix. B.1 and B.3. LipsNet's parameters were adopted from the original paper (Song et al. 2023) and are given in Table B.14.

B.6 Ablation Study

The hyperparameters for each temporal and spatial term used in the ablation study are listed in Table B.15.

C Additional Experimental Results

C.1 Analysis of Predictor Stability

The predictor is used as the learning target for the actor, but because the predictor itself is updated at every step, this introduces a moving target instability. We therefore advocate applying soft updates to the predictor in the SAC setting, and in the PPO setting using more parallel environments along with a smaller λ_P coefficient. To validate these hypotheses, we conducted experiments in two configurations:

- SAC: Only the spatial term of ASAP is applied. We trained two variants—one with soft updates on the predictor and one without—and compared their cumulative

Method	Gymnasium Environments					
	LunarLander	Pendulum	Reacher	Ant	Hopper	Walker
CAPS	$\lambda_T = 0.001$ $\lambda_S = 0.005$ $\sigma = 0.2$	$\lambda_T = 0.01$ $\lambda_S = 0.05$ $\sigma = 0.2$	$\lambda_T = 0.1$ $\lambda_S = 0.5$ $\sigma = 0.2$	$\lambda_T = 0.1$ $\lambda_S = 0.5$ $\sigma = 0.2$	$\lambda_T = 0.1$ $\lambda_S = 0.5$ $\sigma = 0.2$	$\lambda_T = 0.1$ $\lambda_S = 0.5$ $\sigma = 0.2$
L2C2	$\overline{\lambda} = 0.1$ $\underline{\lambda} = 0.001$ $\sigma = 1.0$	$\overline{\lambda} = 1.0$ $\underline{\lambda} = 0.01$ $\sigma = 1.0$	$\overline{\lambda} = 10.0$ $\underline{\lambda} = 0.1$ $\sigma = 1.0$	$\overline{\lambda} = 10.0$ $\underline{\lambda} = 0.1$ $\sigma = 1.0$	$\overline{\lambda} = 10.0$ $\underline{\lambda} = 0.1$ $\sigma = 1.0$	$\overline{\lambda} = 10.0$ $\underline{\lambda} = 0.1$ $\sigma = 1.0$
GRAD	$\lambda_T = 0.001$	$\lambda_T = 0.01$	$\lambda_T = 0.1$	$\lambda_T = 0.1$	$\lambda_T = 0.1$	$\lambda_T = 0.1$
ASAP	$\lambda_T = 0.005$ $\lambda_S = 0.03$ $\lambda_P = 2.0$	$\lambda_T = 0.005$ $\lambda_S = 0.03$ $\lambda_P = 2.0$	$\lambda_T = 0.1$ $\lambda_S = 0.1$ $\lambda_P = 2.0$	$\lambda_T = 0.05$ $\lambda_S = 0.3$ $\lambda_P = 2.0$	$\lambda_T = 0.07$ $\lambda_S = 0.3$ $\lambda_P = 2.0$	$\lambda_T = 0.05$ $\lambda_S = 0.3$ $\lambda_P = 2.0$

Table B.7: Detailed hyperparameters for each method in the Gymnasium-PPO setting.

Method	Gymnasium Environments					
	LunarLander	Pendulum	Reacher	Ant	Hopper	Walker
CAPS	$\lambda_T = 0.1$ $\lambda_S = 0.5$ $\sigma = 0.2$	$\lambda_T = 1.0$ $\lambda_S = 5.0$ $\sigma = 0.2$	$\lambda_T = 0.1$ $\lambda_S = 0.5$ $\sigma = 0.2$	$\lambda_T = 0.1$ $\lambda_S = 0.5$ $\sigma = 0.2$	$\lambda_T = 0.1$ $\lambda_S = 0.5$ $\sigma = 0.2$	$\lambda_T = 0.1$ $\lambda_S = 0.5$ $\sigma = 0.2$
L2C2	$\overline{\lambda} = 10.0$ $\underline{\lambda} = 0.1$ $\sigma = 1.0$	$\overline{\lambda} = 100.0$ $\underline{\lambda} = 1.0$ $\sigma = 1.0$	$\overline{\lambda} = 10.0$ $\underline{\lambda} = 0.1$ $\sigma = 1.0$	$\overline{\lambda} = 10.0$ $\underline{\lambda} = 0.1$ $\sigma = 1.0$	$\overline{\lambda} = 10.0$ $\underline{\lambda} = 0.1$ $\sigma = 1.0$	$\overline{\lambda} = 10.0$ $\underline{\lambda} = 0.1$ $\sigma = 1.0$
GRAD	$\lambda_T = 1.0$	$\lambda_T = 1.0$	$\lambda_T = 1.0$	$\lambda_T = 1.0$	$\lambda_T = 1.0$	$\lambda_T = 1.0$
ASAP	$\lambda_T = 0.5$ $\lambda_S = 10.0$ $\lambda_P = 2.0$	$\lambda_T = 0.5$ $\lambda_S = 5.0$ $\lambda_P = 2.0$	$\lambda_T = 0.5$ $\lambda_S = 5.0$ $\lambda_P = 2.0$	$\lambda_T = 0.5$ $\lambda_S = 10.0$ $\lambda_P = 2.0$	$\lambda_T = 0.5$ $\lambda_S = 10.0$ $\lambda_P = 2.0$	$\lambda_T = 0.5$ $\lambda_S = 5.0$ $\lambda_P = 2.0$

Table B.8: Detailed hyperparameters for each method in the Gymnasium-SAC setting.

returns to assess the effect of stabilizing the moving target.

- PPO: In the Isaac-Lab environment, the number of parallel environments was held fixed while λ_P was varied, and the resulting changes in cumulative return were measured.

Soft Update. Table C.1 compares ASAP in the SAC setting with and without applying soft updates to the predictor. Although both variants used the same λ_S, their behavior diverged: in dynamic environments (Hopper and Walker), removing the soft update yielded higher cumulative returns, whereas in high-precision and persistence-critical tasks (Pendulum and Reacher) it caused performance degradation. In contrast, the version with soft update exhibited stable performance across all tasks, closely matching the original SAC. Removing the soft update produced higher cumulative returns in some environments, but the decline on high-precision tasks indicates a moving target problem. Soft updates effectively alleviate this, stabilizing the predictor and preventing reward degradation in sensitive tasks.

Environment Parallelization. Table C.2 shows the cumulative return and smoothness as a function of λ_P in the PPO setting with many parallel environments. Reach-Franka and Anymal-Velocity-Rough use 4096 environments; all other hyperparameters except λ_P are kept the same as in Appendix B.4. At high λ_P, the jitter in the predictor's output is excessively propagated into the policy updates, causing the policy to lose a consistent learning target and chase a moving target, which results in a large drop in cumulative return. In contrast, reducing λ_P lowers sensitivity to the predictor's instability, stabilizing both cumulative return and smoothness; as shown in Table 4, performance then converges to levels comparable to the original PPO. All experiments were conducted with the same total number of environment interactions across multiple seeds. These results suggest that collecting abundant data from many parallel environments under a fixed policy while decreasing λ_P to mitigate the moving target problem is an effective strategy for obtaining stable performance.

C.2 Correlation between Noise and λ_S

According to Lemma 1, increasing the observation noise σ_ξ enlarges the radius of the similar state distribution $P(\cdot \mid s_{t-1})$, which defines the neighborhood over which the policy's smoothness is enforced. This expansion directly affects

Method	Isaac-Lab Environments			
	Reach-Franka	Lift-Cube-Franka	Repose-Cube-Allegro	Anymal-Velocity-Rough
ASAP	$\lambda_T = 0.1$ $\lambda_S = 1.0$ $\lambda_P = 0.02$	$\lambda_T = 0.05$ $\lambda_S = 0.3$ $\lambda_P = 0.02$	$\lambda_T = 0.01$ $\lambda_S = 0.1$ $\lambda_P = 0.02$	$\lambda_T = 0.05$ $\lambda_S = 0.3$ $\lambda_P = 0.02$

Table B.13: Detailed hyper parameters for each method in the Isaac-PPO setting.

Parameter	Default Value
λ	1e-5
initial K	50
hidden layer in $f(x)$	[256, 256]
hidden layer in $K(x)$	[32]
activation in $f(x)$	ReLU
small constant ϵ	1e-4

Table B.14: Parameters for LipsNet setting

	Hopper	Walker
CAPS-λ_T	0.07	0.05
GRAD-λ_T	0.07	0.05
CAPS-λ_S	0.5	0.5
L2C2-λ_S	10.0	10.0
ASAP-λ_S	0.3	0.3

Table B.15: Hyper parameters for each temporal and spatial term used in the ablation study on Hopper and Walker.

the composite Lipschitz upper bound in Theorem 1, as the term

$$d_A\Big(f\circ T(s_{t-1}, a_{t-1}, \xi_{t-1}^{(1)}),\ f\circ T(s_{t-1}, a_{t-1}, \xi_{t-1}^{(2)})\Big) \tag{A.2-6}$$

is bounded by $K_{\text{comp}} \cdot 2\sigma_\xi$, where $2\sigma_\xi$ reflects the diameter of the similar state neighborhood.

As σ_ξ increases, this neighborhood grows, and the regularization term

$$L = \left\|\pi_\phi(s_t) - \mathbb{E}_{\tilde{s}_t\sim P(\cdot|s_{t-1})}\left[\pi_\phi(\tilde{s}_t)\right]\right\|_2^2 \tag{A.2-7}$$

penalizes discrepancies of the policy output across this neighborhood. However, as the neighborhood expands, the loss begins to impose constraints that resemble a global Lipschitz condition rather than a local one. Enforcing such a global constraint may inadvertently suppress necessary policy variations, thus impairing expressiveness.

To mitigate this effect, the coefficient λ_S should be adjusted inversely with respect to σ_ξ, reducing the regularization strength as the effective neighborhood enlarges. This adjustment ensures that the spatial smoothness constraint remains locally adaptive, preventing over-smoothing while maintaining robustness to observation noise.

To validate this, we conducted experiments in the Isaac-Lab Isaac-Repose-Cube-Allegro-v0 environment, comparing original PPO and ASAP with $\lambda_S = 0.1$ and $\lambda_S = 0.01$ under observation noise scaled by factors of 1.0, 1.5, and 2.0. Each configuration was trained with 5 random seeds and evaluated over 50 distinct seeds. The corresponding empirical results are shown in Figures C.1 and C.2.

Environment	W/O Soft Update		W/ Soft Update	
	re↑	*sm*↓	*re*↑	*sm*↓
Pendulum	-147.2 (69.3)	0.295 (0.109)	-145.8 (68.8)	0.329 (0.099)
Reacher	-3.87 (1.43)	0.042 (0.014)	-3.64 (1.40)	0.045 (0.016)
Hopper	3490 (68)	0.394 (0.044)	3440 (215)	0.511 (0.105)
Walker	4811 (387)	0.606 (0.099)	4503 (229)	0.685 (0.083)

Table C.1: Cumulative return (*re*) and smoothness score (*sm*) of for ASAP with and without soft updates to the predictor.

Environment	Reach-Franka		Velocity-Rough	
	re↑	*sm*↓	*re*↑	*sm*↓
ASAP(λ_P2.0)	0.331 (0.253)	0.723 (0.057)	14.63 (4.43)	2.506 (0.456)
ASAP(λ_P0.2)	0.419 (0.222)	0.695 (0.079)	15.69 (3.23)	2.814 (0.328)
ASAP(λ_P0.02)	0.525 (0.195)	0.658 (0.065)	16.09 (2.92)	2.861 (0.306)

Table C.2: Cumulative return (*re*) and smoothness score (*sm*) of ASAP for $\lambda_P \in \{2.0, 0.2, 0.02\}$.

The results show that for original PPO and ASAP with $\lambda_S = 0.01$, there is little to no observable correlation between the level of observation noise and either the smoothness score or cumulative return. In contrast, ASAP with $\lambda_S = 0.1$ exhibits a slight decrease in cumulative return and a substantial degradation in smoothness score as observation noise increases. These findings indicate a dependency between the magnitude of observation noise and the spatial regularization coefficient λ_S, suggesting that λ_S should be adjusted in accordance with the noise level.

C.3 Analysis of High Variance in Cumulative Return in IsaacLab Experiments

In the main text, the high variance of ASAP's cumulative return in Table 4 is explained as a phenomenon where the oscillation suppression process constrains exploration, causing insufficient learning in some random seeds. To validate this,

Method	Reach-Franka		Lift-Cube-Franka		Repose-Cube-Allegro		Anymal-Velocity-Rough	
	re ↑	*sm* ↓	*re* ↑	*sm* ↓	*re* ↑	*sm* ↓	*re* ↑	*sm* ↓
PPO Base	0.529 (0.226)	0.904 (0.098)	147.1 (10.1)	1.437 (0.428)	**40.34** (15.12)	2.702 (0.705)	**15.49** (4.18)	3.464 (0.379)
ASAP	**0.655** (0.183)	**0.638** (0.055)	**150.1** (6.3)	**0.714** (0.054)	37.316 (14.14)	**2.542** (0.521)	14.89 (4.89)	**2.620** (0.428)

Table C.3: Cumulative return (*re*) and smoothness score (*sm*) for PPO Base and PPO + ASAP on the Isaac-Lab environment, evaluated using the best three checkpoints. Higher *re* and lower *sm* are better. Bold indicates the best per environment. Standard deviations shown in parentheses.

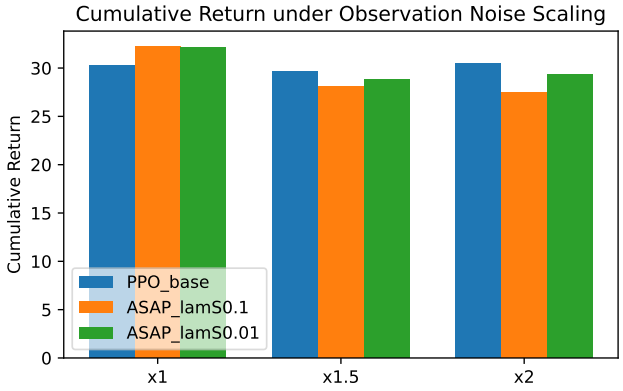


Figure C.1: Cumulative returns of original PPO and ASAP (with $\lambda_S = 0.1$ and $\lambda_S = 0.01$) under noise multiplier settings in the Repose-Cube-Allegro environment.

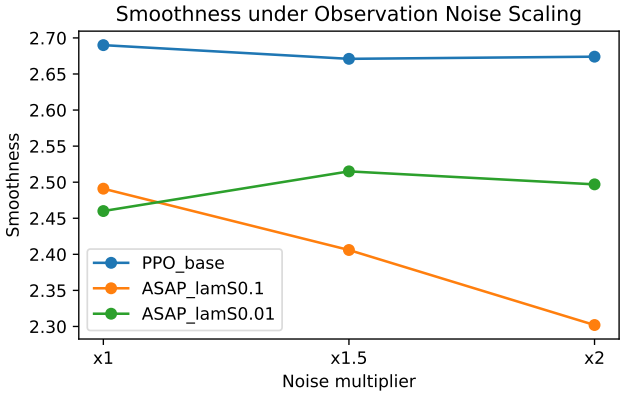


Figure C.2: Smoothness score of original PPO and ASAP (with $\lambda_S = 0.1$ and $\lambda_S = 0.01$) under noise multiplier settings in the Repose-Cube-Allegro environment.

we selected the top three checkpoints with the highest final return and re-evaluated their performance.

Table C.3 shows the experimental results. Except for the velocity environment, these top-3 weights exhibited cumulative return variance that was comparable to or lower than that of the original PPO, while retaining peak performance. In the velocity case, however, even after selecting the best-performing weights, both original PPO and ASAP showed decreased cumulative return and increased variance relative to the values reported in Table 4, indicating reduced reliability. This behavior may result from the interaction between the high environment diversity inherent in velocity-rough and the evaluation setup in which exploration is suppressed.

These observations suggest that although ASAP occasionally produces outliers with large drops in cumulative return compared to original PPO, its best-case performance remains intact; such weakness can be alleviated by multi-seed best-of-N selection or more robust checkpoint selection strategies.

D Additional Visualization

D.1 Franka Reacher

In the Franka reach task within the Isaac-Lab environment, ASAP demonstrated strong performance by improving both cumulative return (*re*) and smoothness score (*sm*). Notably, *sm* was improved by 31.3%, and the mean error between the end-effector and the target decreased by 30.3%, from 0.90 cm to 0.62 cm. This indicates that smoothing constraints are particularly beneficial for tasks that demand high precision and sustained stability. Figure D.1 illustrates the mean end-effector-to-target error for both the original PPO and our approach.

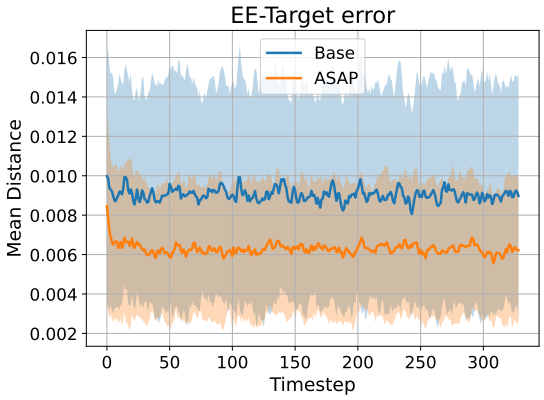
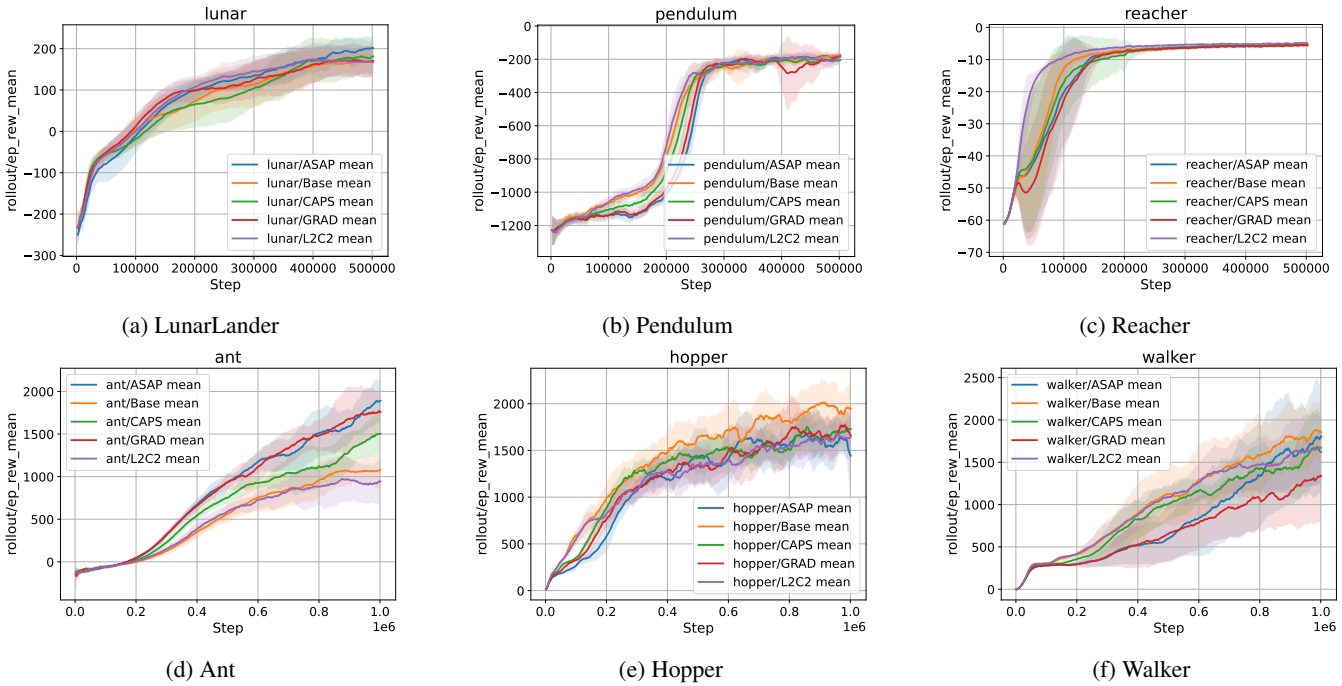


Figure D.1: Distance between the end-effector and the target in the Franka reach task. Solid line denotes the mean; shaded region represents standard deviation.

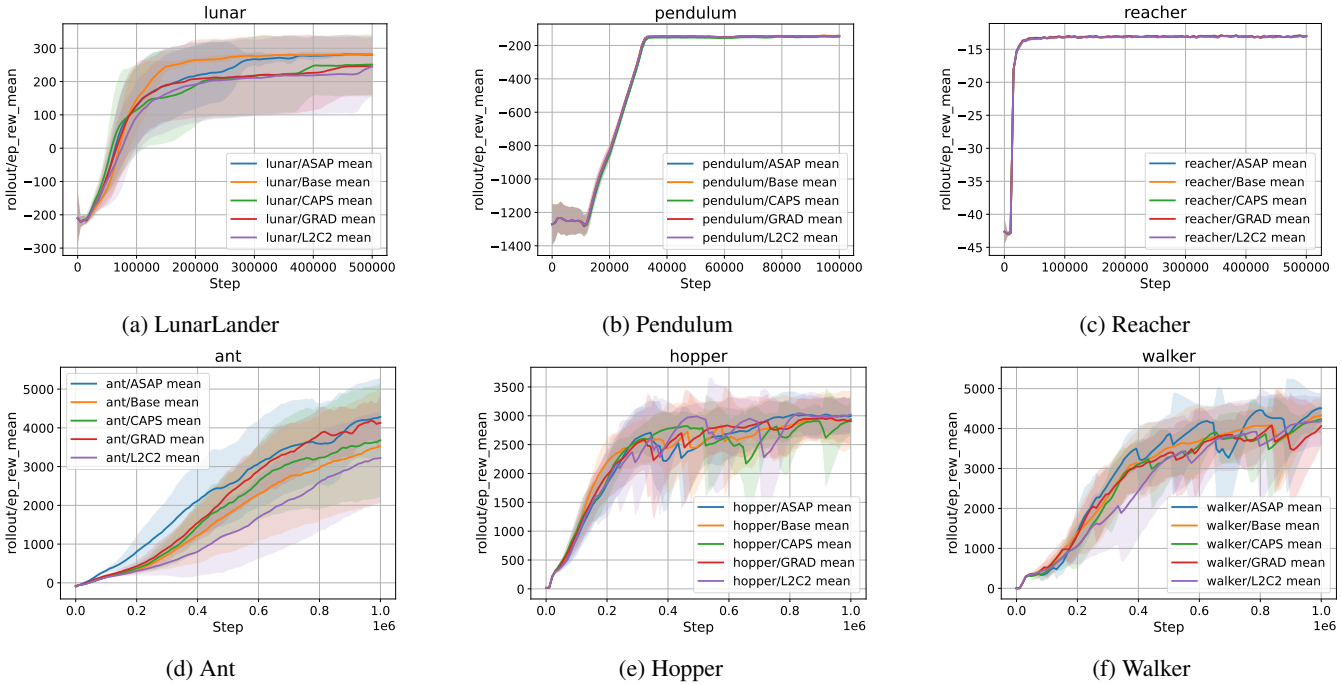
D.2 Learning Curve



(a) LunarLander (b) Pendulum (c) Reacher

(d) Ant (e) Hopper (f) Walker

Figure D.2: Learning curves for Gymnasium-PPO across six environments; solid lines denote the mean and shaded regions represent standard deviation.

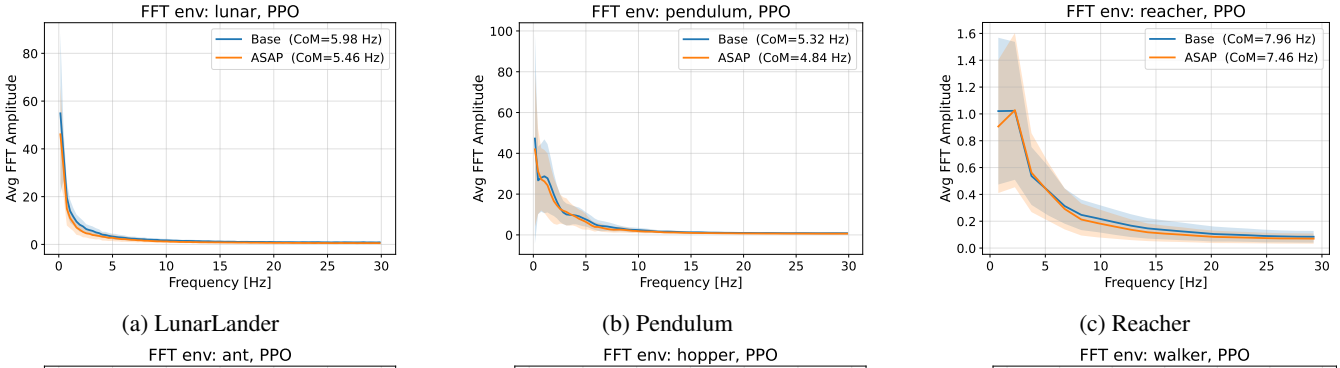


(a) LunarLander (b) Pendulum (c) Reacher

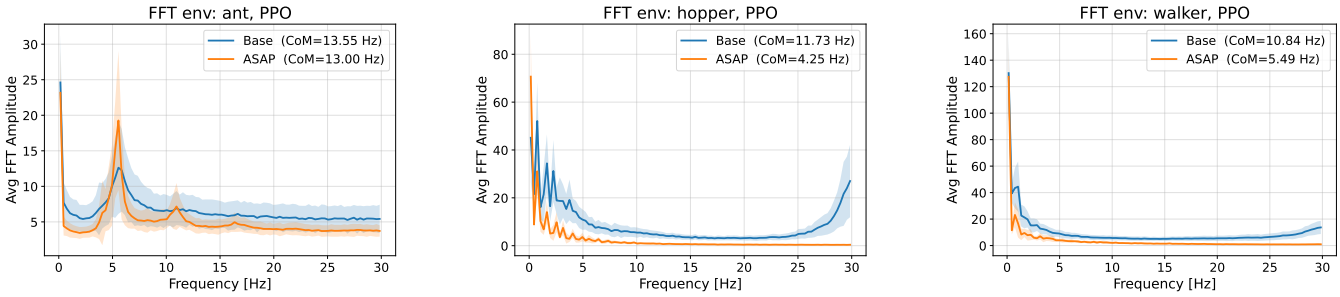
(d) Ant (e) Hopper (f) Walker

Figure D.3: Learning curves for Gymnasium-SAC across six environments; solid lines denote the mean and shaded regions represent standard deviation.

D.3 Frequency Spectrum of Action

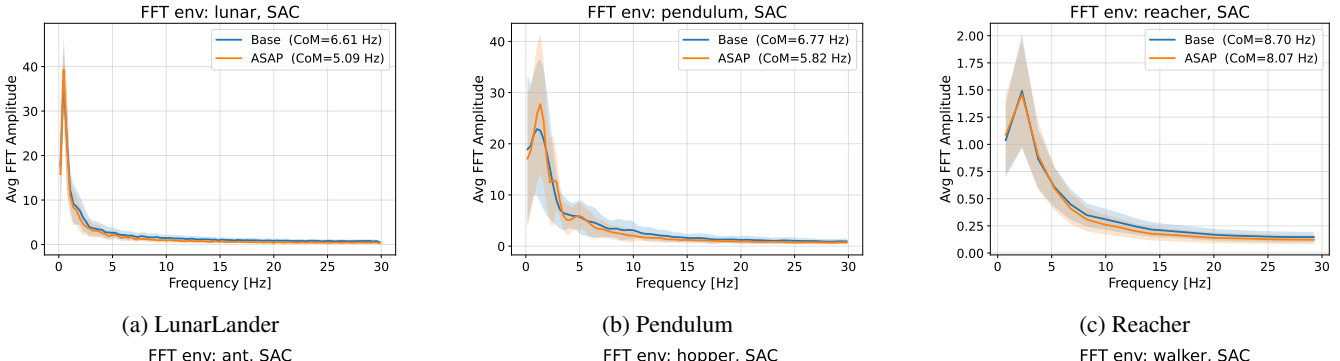


(a) LunarLander (b) Pendulum (c) Reacher

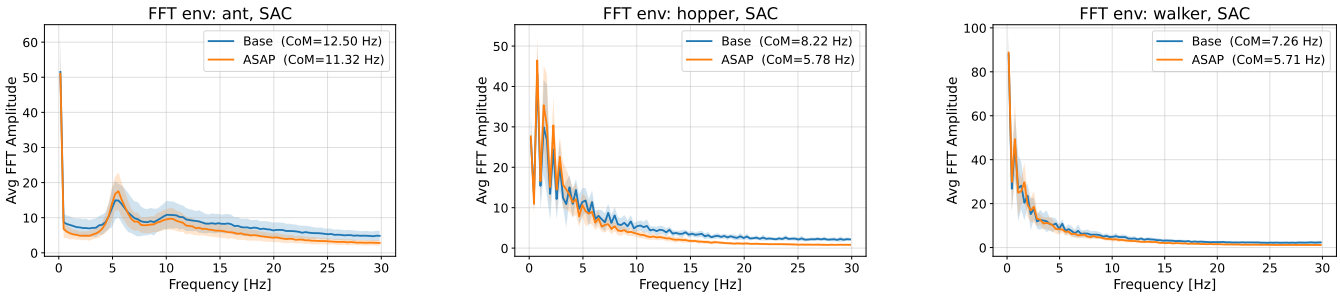


(d) Ant (e) Hopper (f) Walker

Figure D.4: Frequency spectrum of actions for Gymnasium-PPO across six environments; solid lines denote the mean and shaded regions represent standard deviation.



(a) LunarLander (b) Pendulum (c) Reacher



(d) Ant (e) Hopper (f) Walker

Figure D.5: Frequency spectrum of actions for Gymnasium-PPO across six environments; solid lines denote the mean and shaded regions represent standard deviation.

References

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. Pmlr.

Kobayashi, T. 2022. L2c2: Locally lipschitz continuous constraint towards stable and smooth reinforcement learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4032–4039. IEEE.

Lee, I.; Cao, H.-G.; Dao, C.-T.; Chen, Y.-C.; and Wu, I.-C. 2024. Gradient-based Regularization for Action Smoothness in Robotic Control with Reinforcement Learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 603–610. IEEE.

Makoviichuk, D.; and Makoviychuk, V. 2021. rl-games: A High-performance Framework for Reinforcement Learning. https://github.com/Denys88/rl_games.

Mittal, M.; Yu, C.; Yu, Q.; Liu, J.; Rudin, N.; Hoeller, D.; Yuan, J. L.; Singh, R.; Guo, Y.; Mazhar, H.; Mandlekar, A.; Babich, B.; State, G.; Hutter, M.; and Garg, A. 2023. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. *IEEE Robotics and Automation Letters*, 8(6): 3740–3747.

Mysore, S.; Mabsout, B.; Mancuso, R.; and Saenko, K. 2021. Regularizing action policies for smooth control with reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 1810–1816. IEEE.

Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; and Dormann, N. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268): 1–8.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Song, X.; Duan, J.; Wang, W.; Li, S. E.; Chen, C.; Cheng, B.; Zhang, B.; Wei, J.; and Wang, X. S. 2023. LipsNet: A smooth and robust neural network with adaptive Lipschitz constant for high accuracy optimal control. In *International Conference on Machine Learning*, 32253–32272. PMLR.

Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J. U.; De Cola, G.; Deleu, T.; Goulao, M.; Kallinteris, A.; Krimmel, M.; KG, A.; et al. 2024. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*.