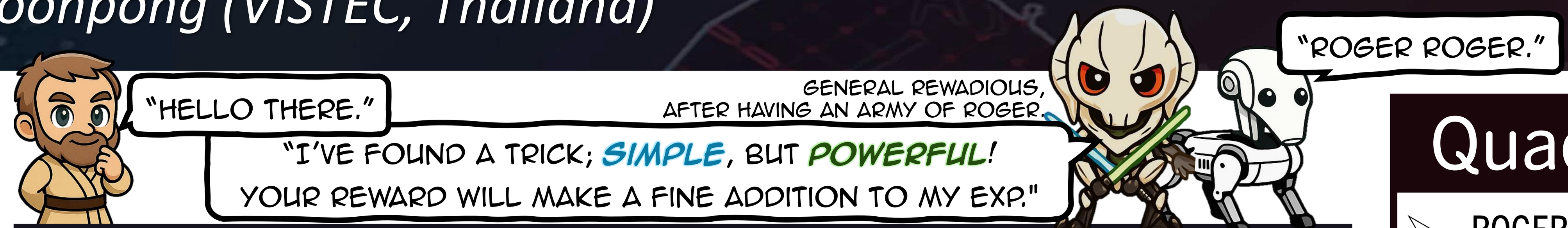
Gain Tuning Is Not What You Need:

Reward Gain Adaptation for Constrained Locomotion Learning

Arthicha Srisuchinnawong and Poramate Manoonpong (VISTEC, Thailand)

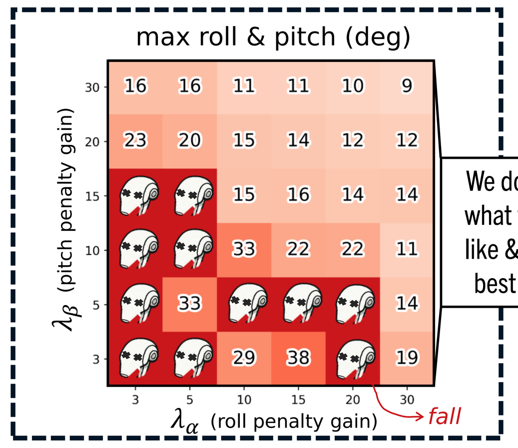
BRAIN VISTEC
ROBOTICS SCIENCE AND SYSTEMS

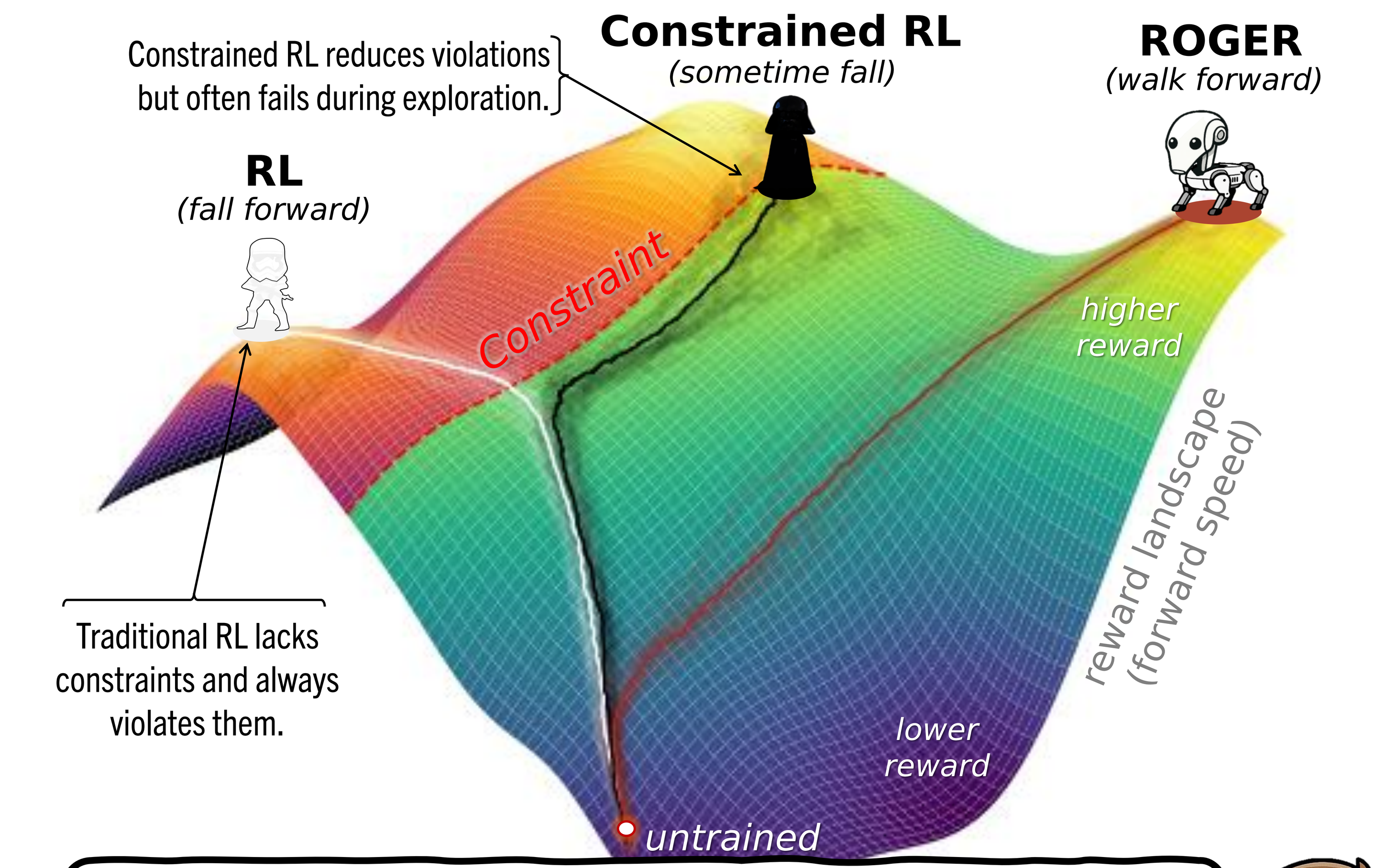


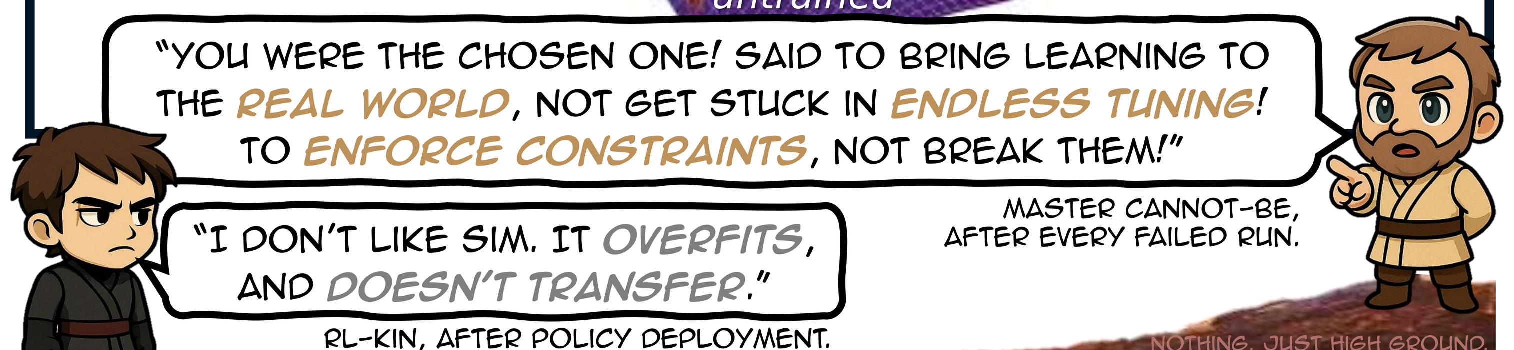
Introduction

Real-world locomotion learning remains extremely challenging due to:

- **Sample inefficiency**, which makes training very time-consuming and unstable under limited samples.
- **Constraint and safety violations**, especially during learning.
- **Exhaustive and unpredictable reward tuning**, which relies on trial and error with no clear way to ensure constraint satisfaction.







Constrained RL

Previous works either use:

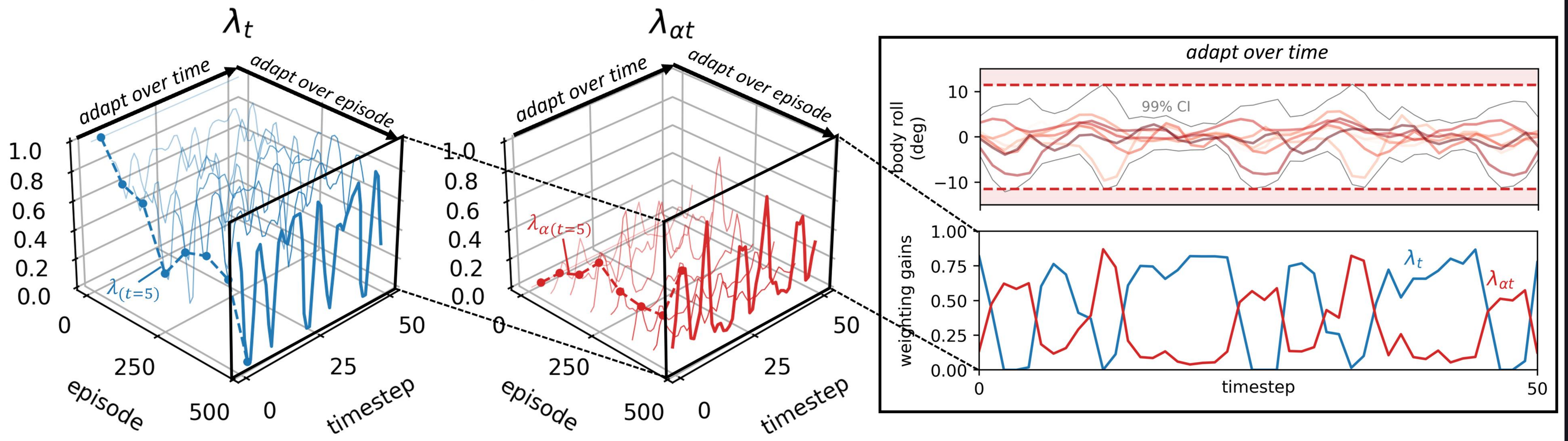
- **Fixed weighting gains** (e.g., carefully tuned reward functions, CBFs) ⟶ require extensive tuning, still risk constraint violations, and typically rely on deploying a frozen policy after training; or
- **Adaptive weighting gains** (e.g., PDO, OL-AUX, CRPO) ⟶ suffer from update delays, increased computation, and/or often fail to enforce constraints during learning.

Techniques	Reward Gain (λ_t)	Penalty Gain (λ_{it})
Fixed-weighting & Control Barrier Function (CBF)	1.0 (fixed)	tuned and fixed
Primal-Dual Optimization (PDO)	1.0 (fixed)	$[\lambda_{it} + \eta_\lambda(\tilde{R}_{it} - (\tau_i - \delta_i))]_+$
Online Learning of Auxiliary Loss (OL-AUX)	1.0 (fixed)	$\lambda_{it} + \eta_\lambda \nabla\mathbb{E}[R_{vt}]\nabla\mathbb{E}[\tilde{R}_{it}]$
Constrained Rectified Policy Optimization (CRPO)	0.0 if (exist $\tilde{R}_{it} > \tau_i - \delta_i$) else 1.0	1.0 if ($\tilde{R}_{it} > \tau_i - \delta_i$) else 0.0

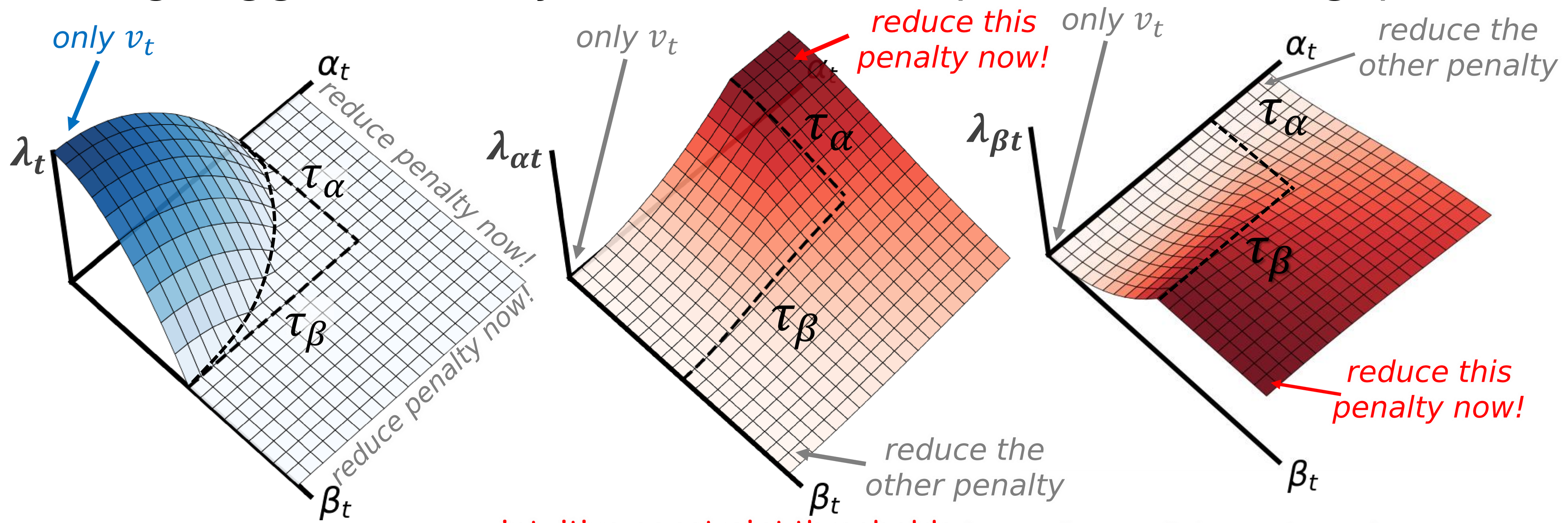
ROGER *(Reward-Oriented Gains via Embodied Regulation)*

ROGER adaptively regulates penalty weights in real-time using embodied feedback and intuitive constraint thresholds.

- **Key idea:** As penalties (e.g., body orientation) approach their thresholds, ROGER increasingly prioritizes minimizing those penalties. Once the robot has learned safe behavior, the focus gradually shifts back to optimizing the primary objective (i.e., forward speed).



- **How it works:** ROGER employs fixed manifolds to map current penalty values to weighting gains on-the-fly, both at each timestep and across learning episodes.



intuitive constraint thresholds (max roll, max pitch, max torque)

$$\delta_{\alpha t} = \frac{|\alpha_t|}{\boxed{\tau_\alpha}}, \quad \delta_{\beta t} = \frac{|\beta_t|}{\boxed{\tau_\beta}}$$

constraint proximities

$$\Delta_t = min\{\delta_{\alpha t}^2 + \delta_{\beta t}^2, 1.0\}, \quad r_{\alpha t} = (\delta_{\alpha t}^2)/(\delta_{\alpha t}^2 + \delta_{\beta t}^2), \quad r_{\beta t} = (\delta_{\beta t}^2)/(\delta_{\alpha t}^2 + \delta_{\beta t}^2)$$

penalty ratios

$$\begin{bmatrix}\lambda_t \\ \lambda_{\alpha t} \\ \lambda_{\beta t}\end{bmatrix} = \begin{bmatrix}1 - \Delta_t \\ r_{\alpha t}\Delta_t \\ r_{\beta t}\Delta_t\end{bmatrix}$$

adaptive weighting gains

- ROGER is **Lyapunov-stable** and guarantees improvement of the primary objective/reward after learning.
- Unlike learning-based methods (e.g., PDO, OL-AUX), ROGER introduces **no update delay**, requires **no tuning**, and has **low computation** time (~0.46 ms).

Algorithm : Reinforcement Learning with ROGER

1: Perform exploration and collect trajectory τ.

2: Compute estimated penalties ($|\tilde{\alpha}_t|$ and $|\tilde{\beta}_t|$) using: $\begin{bmatrix}|\tilde{\alpha}_t| \\ |\tilde{\beta}_t|\end{bmatrix} = \sum \gamma^i \begin{bmatrix}|\alpha_{t+i}| \\ |\beta_{t+i}|\end{bmatrix} / \sum \gamma^i$.

3: Compute weighting gains (λs) using ROGER: $\begin{bmatrix}\lambda_t \\ \lambda_{\alpha t} \\ \lambda_{\beta t}\end{bmatrix} = \begin{bmatrix}1 - \Delta(|\tilde{\alpha}_t|, |\tilde{\beta}_t|) \\ r_{\alpha t}\,\Delta(|\tilde{\alpha}_t|, |\tilde{\beta}_t|) \\ r_{\beta t}\,\Delta(|\tilde{\alpha}_t|, |\tilde{\beta}_t|)\end{bmatrix}$.

4: Update policy using combined reward or advantage:

$$R_t = \lambda_t v_t - \lambda_{\alpha t}|\alpha_t| - \lambda_{\beta t}|\beta_t| \quad \text{or} \quad A_t = \lambda_t A_{vt} - \lambda_{\alpha t} A_{\alpha t} - \lambda_{\beta t} A_{\beta t}$$

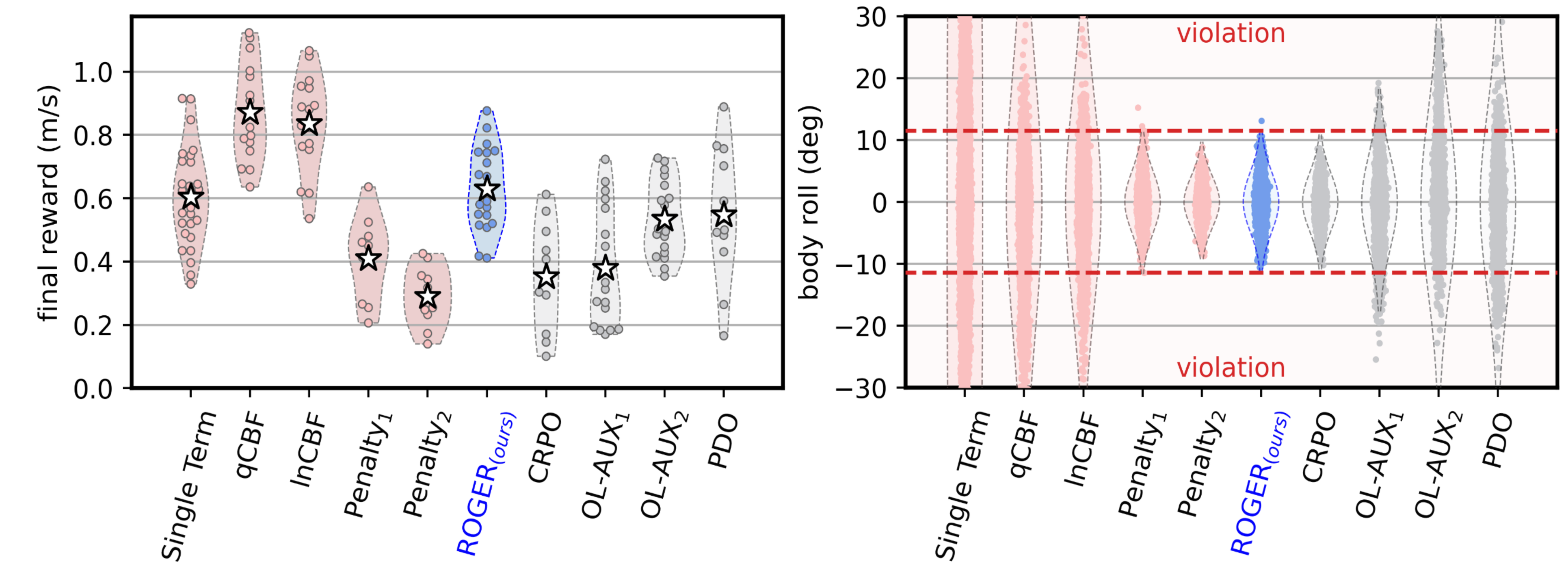
ROGER-

– POORLY TUNED RL

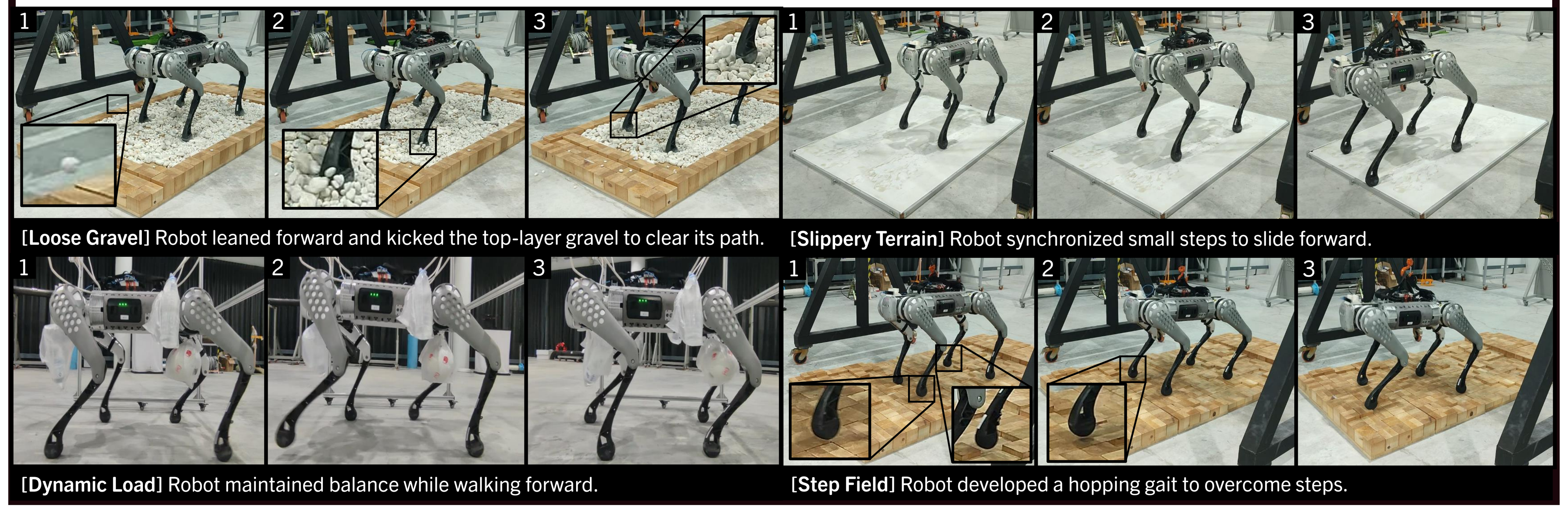
WELL-TUNED CONSTRAINED RL –

Quadruped Robot

- ROGER achieves constraint satisfaction with **<5 violations over 50,000 training timesteps** even when using small training samples (only ≈50 timesteps/update).
- ROGER yields **50% higher final reward** than state-of-the-art methods with near-zero violations.



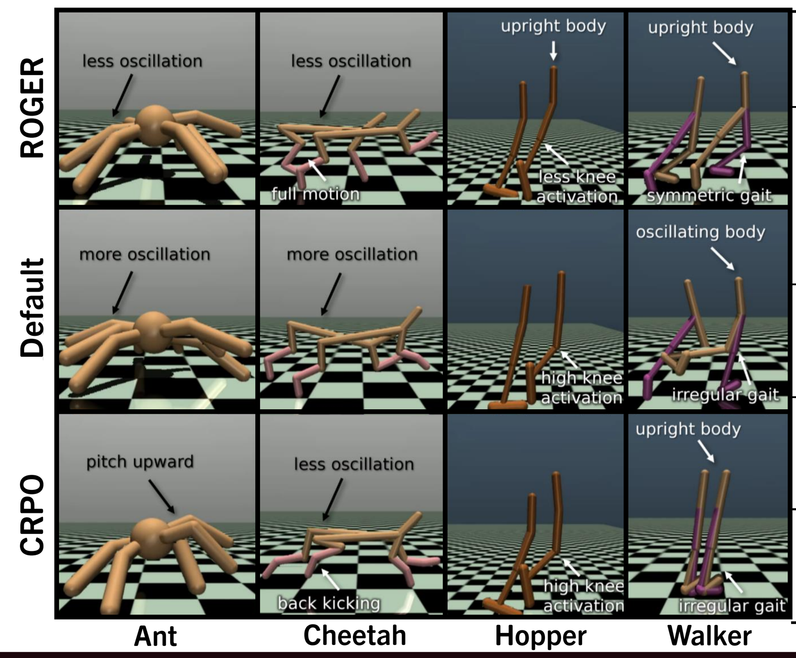
- **Physical locomotion learning** was achieved **within 30 mins from scratch without any falls** on challenging conditions (loose gravel, step field, slippery terrain, and with dynamic load).



[Loose Gravel] Robot leaned forward and kicked the top-layer gravel to clear its path.
[Slippery Terrain] Robot synchronized small steps to slide forward.
[Dynamic Load] Robot maintained balance while walking forward.
[Step Field] Robot developed a hopping gait to overcome steps.

MuJoCo Benchmark

- ROGER can also be applied to **other types of robots and control**, e.g., FCNN + PPO.
- Under **restricted constraints**, ROGER prioritizes reducing penalties first and then optimizes the primary objective once the constraints are satisfied.



Robot/Environment		Final Values: Default	Final Values: CRPO	Final Values: ROGER	#Violation (throughout): Default	#Violation (throughout): CRPO	#Violation (throughout): ROGER
Ant	Distance (m; ∝ reward)	106.02	57.50	**117.45**	n/a	n/a	n/a
	Torque (Nm), $\tau = 1.0$	**0.25**	0.93	0.39	10^{-10}	0.82	10^{-6}
	Height (m), $\tau = 0.2$	0.00	0.00	0.00	10^{-5}	10^{-7}	10^{-5}
	Roll (°), $\tau = 45$	12.37	16.04	**10.00**	10^{-4}	0.02	10^{-5}
	Pitch (°), $\tau = 45$	8.27	10.88	**5.35**	10^{-8}	10^{-8}	10^{-7}
Cheetah	Distance (m; ∝ reward)	46.10	82.48	**92.25**	n/a	n/a	n/a
	Torque (Nm), $\tau = 1.0$	0.76	0.98	**0.55**	10^{-12}	0.88	10^{-5}
	Pitch (°), $\tau = 10$	6.13	**3.66**	5.53	0.08	10^{-5}	4.7
Hopper	Distance (m; ∝ reward)	5.23	6.42	**6.57**	n/a	n/a	n/a
	Torque (Nm), $\tau = 1.0$	0.78	0.80	**0.33**	0.01	0.01	10^{-7}
	Pitch (°), $\tau = 10$	5.10	4.38	**2.06**	0.14	0.01	10^{-11}
Walker	Distance (m; ∝ reward)	5.99	1.91	**11.65**	n/a	n/a	n/a
	Torque (Nm), $\tau = 1.0$	0.91	0.94	**0.40**	10^{-7}	0.01	10^{-8}
	Pitch (°), $\tau = 45$	21.00	**7.62**	8.31	0.07	8.07	10^{-5}

blue text: best performance (highest distance, lowest penalty); red text: %violation > 0.01

In conclusion, use ROGER when:
- You're doing **high-stakes real-world fine-tuning** or **continual learning**.
- You're **tired of reward tuning**.

Video
GitHub

"TRY OR TRY NOT, THERE IS NO DUE."
DA'YO, AFTER YOU READ THIS POSTER.
https://github.com/Arthicha/ROGER_ROGER_public

References
[Constrained RL] A review of safe reinforcement learning: methods, theories, and applications, PAMI, 2024.
[CRPO] Crpo: a new approach for safe reinforcement learning with convergence guarantee, ICML, 2021.
[OL-AUX] Adaptive auxiliary task weighting for reinforcement learning, NeurIPS, 2019.
[SME-AGOL] Interpretable neural control with adaptable online learning for sample efficient, TNNLS, 2025.
[GOLLUM] Growable neural control with online learning for continual locomotion learning, IJRR, 2025.