



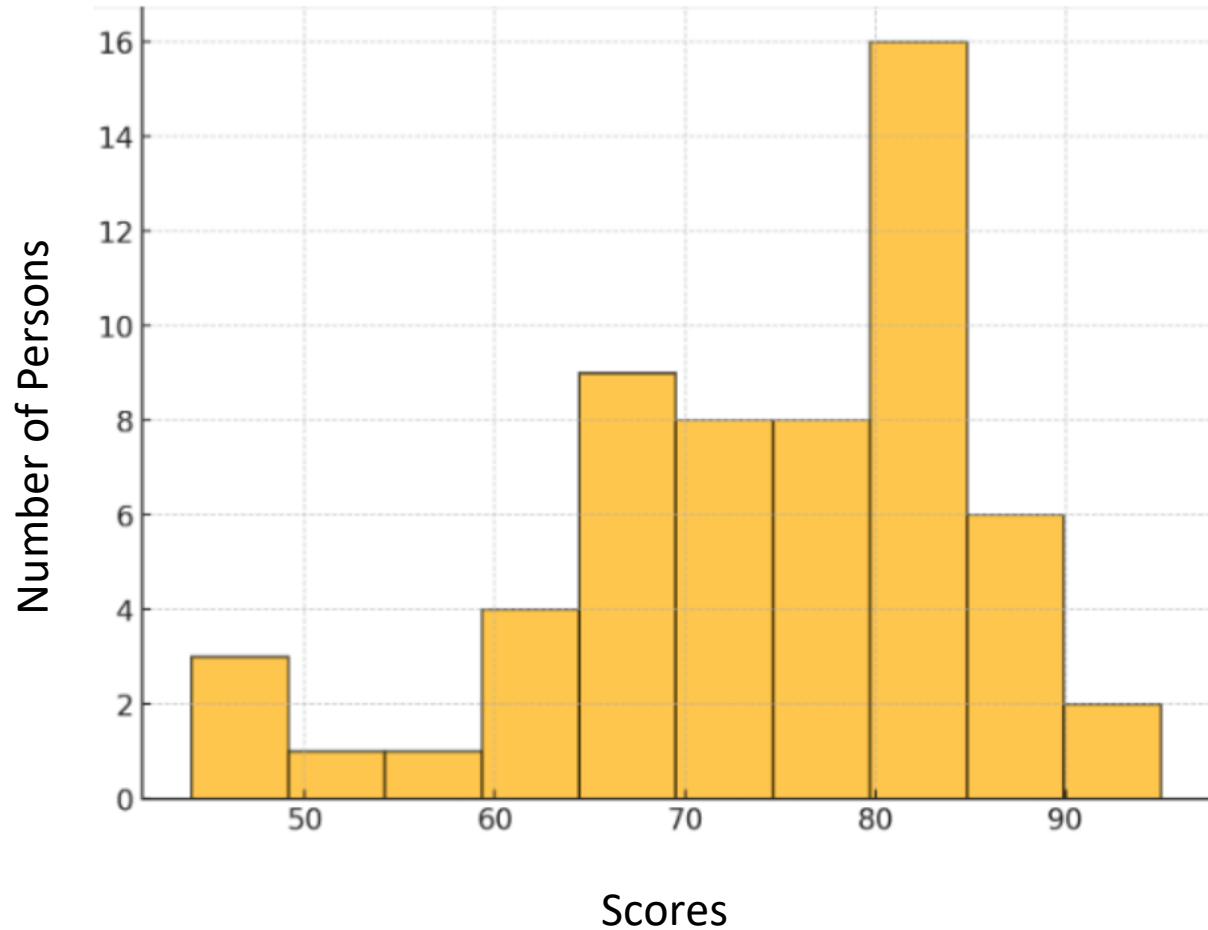
Introduction to Embodied AI

Lecture 11 – Robotic Task I

Prof. He Wang



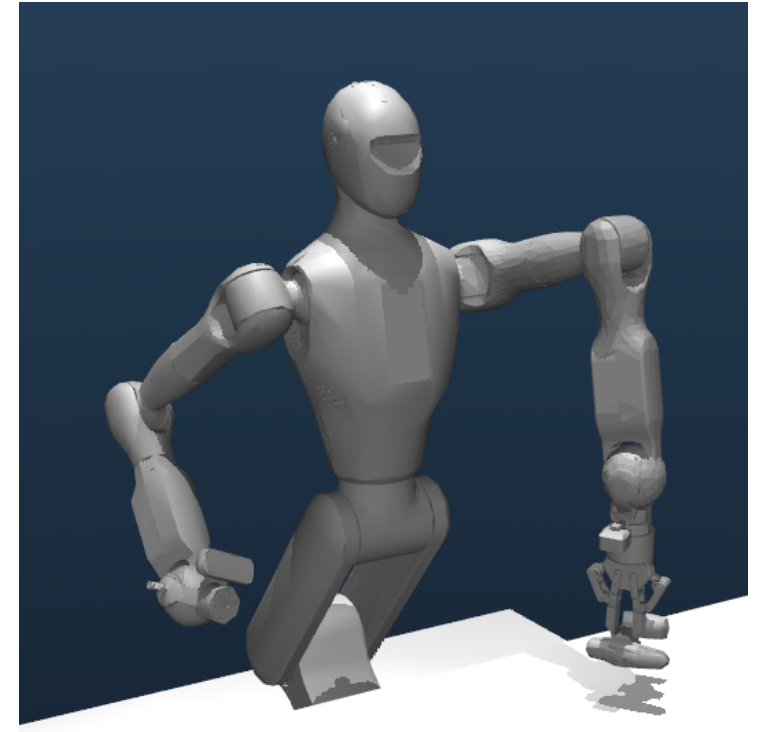
Logistics: Midterm Statistics



- Mean: 74.2
- Median: 76.0
- Highest: 95

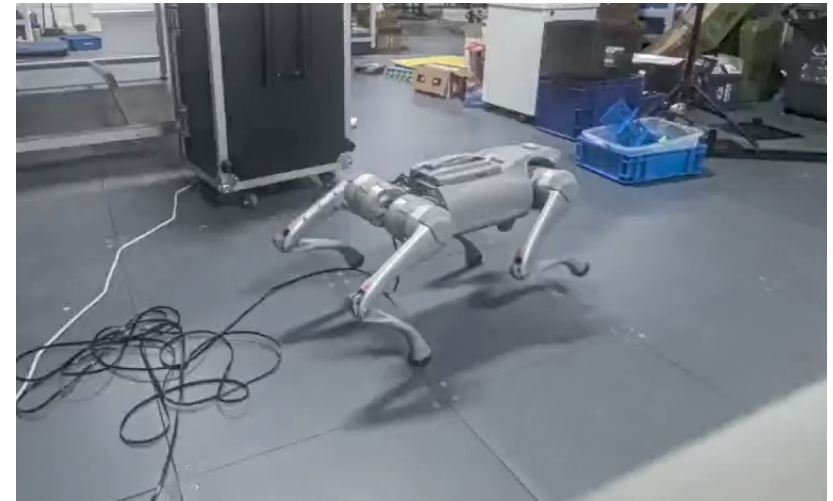
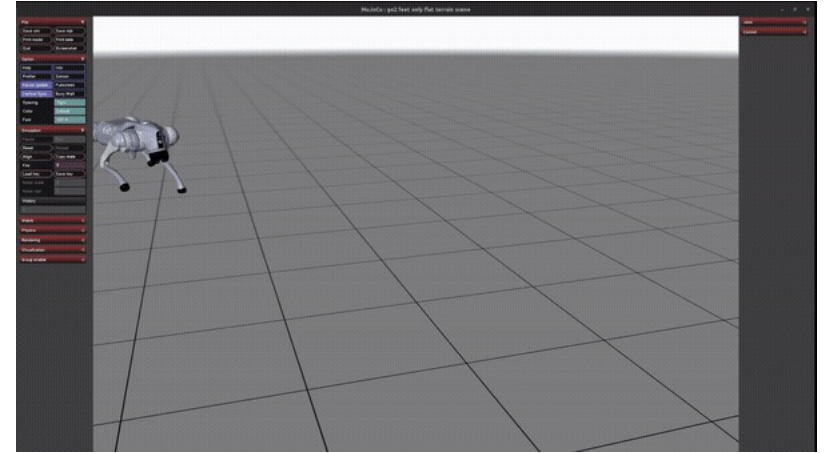
Logistics: Assignment 2

- Object Pose Estimation from Point Cloud Observations
 - Approach 1: direct regression
 - Approach 2: fitting
- Motion plan to grasp
- Each student needs to do this part independently, real experiments will be done in groups.
- Release date: 4/14 Monday
- Due date: 4/30 Wednesday 11:59PM (tonight)



Assignment 3: Quadrupedal Locomotion

- Quadrupedal Locomotion using Unitree Go-2
 - Task 1: Implement your PPO
 - Task II: Quadrupedal Robots – Stand Up
 - Task III: Quadrupedal Robots – Move
- Each student needs to do this part independently, real experiments will be done in groups.
- Release date: 4/30 Wednesday (tonight)
- Due date: 5/14 Wednesday 11:59PM

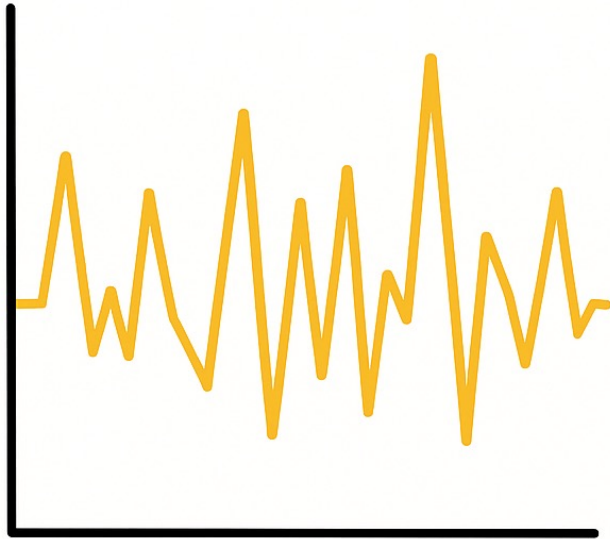


Final Project

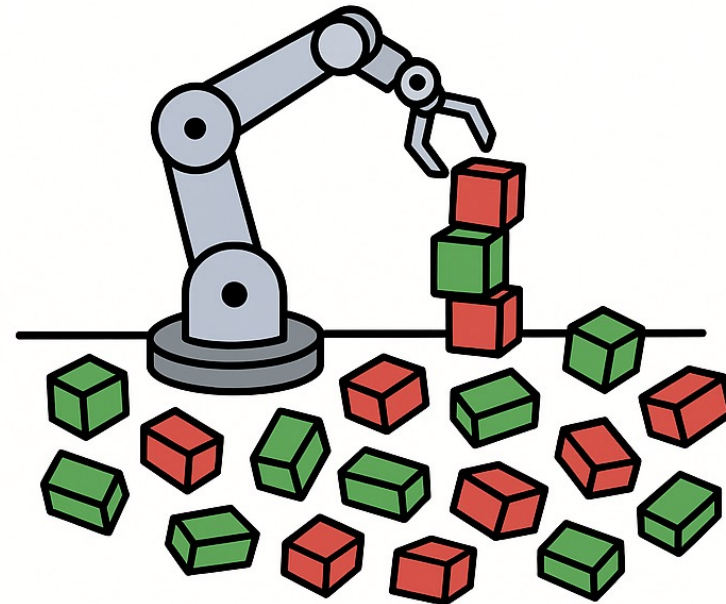
- Quadrupedal Locomotion using Unitree Go-2
- Galbot Pick and Place
- Galbot transfer the object to the box on the back of Go-2
-
- Release date: 5/14 Wednesday 11:59PM
-
- Real experiments will be done in groups.
- Temporary date: 6/2 Wednesday 3:10 – 6:00PM

Problems in RL

policy gradient
is noisy



needs extremely
large numbers of
samples



Off-Policy RL

Policy Gradient is On-policy

$$\theta^* = \arg \max_{\theta} J(\theta)$$

$$J(\theta) = E_{\tau \sim p_{\theta}(\tau)}[r(\tau)]$$

$$\nabla_{\theta} J(\theta) = \underbrace{E_{\tau \sim p_{\theta}(\tau)}}_{\text{this is trouble...}} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]$$

- Neural networks change only a little bit with each gradient step
- On-policy learning can be extremely inefficient!

REINFORCE algorithm:

- 
1. sample $\{\tau^i\}$ from $\pi_{\theta}(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
 2. $\nabla_{\theta} J(\theta) \approx \sum_i \left(\sum_t \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t^i|\mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
 3. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$

can't just skip this!

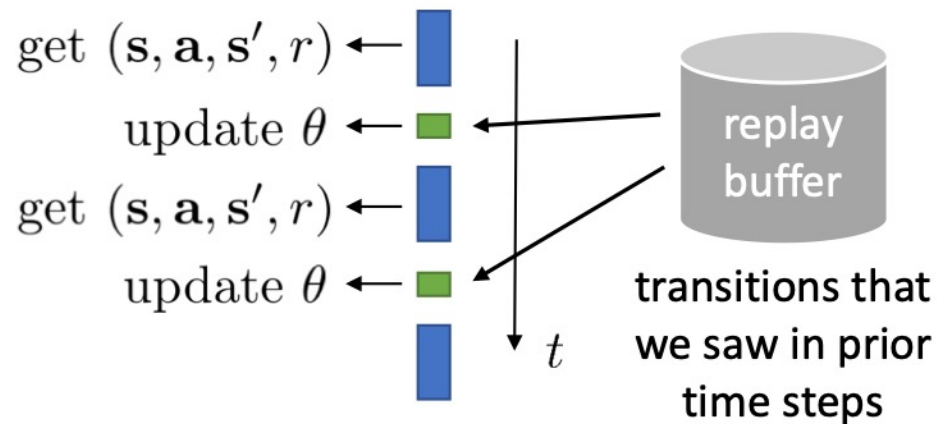


Can we remove the on-policy assumption in AC?

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_{\theta}(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
 2. update $\hat{V}_{\phi}^{\pi}$ using target $r + \gamma \hat{V}_{\phi}^{\pi}(\mathbf{s}')$
 3. evaluate $\hat{A}^{\pi}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}_{\phi}^{\pi}(\mathbf{s}') - \hat{V}_{\phi}^{\pi}(\mathbf{s})$
 4. $\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}|\mathbf{s}) \hat{A}^{\pi}(\mathbf{s}, \mathbf{a})$
 5. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$
- form a **batch** by using old previously seen transitions

off-policy actor-critic



What if we use off-policy data in AC?

online actor-critic algorithm:

- 
1. take action $\mathbf{a} \sim \pi_{\theta}(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
 2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
 3. update $\hat{V}_{\phi}^{\pi}$ using targets $y_i = r_i + \gamma \hat{V}_{\phi}^{\pi}(\mathbf{s}'_i)$ for each $\mathbf{s}_i$
 4. evaluate $\hat{A}^{\pi}(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}_{\phi}^{\pi}(\mathbf{s}'_i) - V_{\phi}^{\pi}(\mathbf{s}_i)$
 5. $\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_i \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^{\pi}(\mathbf{s}_i, \mathbf{a}_i)$
 6. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$
- not the action π_{θ} would have taken!
- not the right target value



$$\mathcal{L}(\phi) = \frac{1}{N} \sum_i \left\| \hat{V}_{\phi}^{\pi}(\mathbf{s}_i) - y_i \right\|^2$$

batch size

This algorithm is broken!

Can you spot the problems?

Fixing the value function

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{V}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i)$ for each $\mathbf{s}_i$
4. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i) - V_\phi^\pi(\mathbf{s}_i)$
5. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

not the right target value

not the action π_θ would have taken!

where does this come from?

3. update $\hat{Q}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i)$ for each $\mathbf{s}_i, \mathbf{a}_i$
 $= r_i + \gamma \hat{Q}_\phi^\pi(\mathbf{s}'_i, \mathbf{a}'_i)$

$\mathbf{a}$ depends on π , $\mathbf{s}'$ also depends on π !

$$V^\pi(s) = \mathbb{E}_{\substack{\mathbf{a} \sim \pi \\ \mathbf{s}' \sim P}} [r(s, \mathbf{a}) + \gamma V^\pi(\mathbf{s}')]$$

~~$$V^\pi(\mathbf{s}_t) = \sum_{t'=t}^T \mathbb{E}_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t]$$~~

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T \mathbb{E}_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$$

“total reward we get if we take $\mathbf{a}_t$ in $\mathbf{s}_t$...
... and then follow the policy π ”

$$\mathcal{L}(\phi) = \frac{1}{N} \sum_i \left\| \hat{Q}_\phi^\pi(\mathbf{s}_i, \mathbf{a}_i) - y_i \right\|^2$$

Only $\mathbf{a}'$ depends on π !

$$Q^\pi(s, a) = \mathbb{E}_{\mathbf{s}' \sim P} \left[r(s, a) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi} [Q^\pi(\mathbf{s}', \mathbf{a}')] \right]$$

Fixing the policy update

online actor-critic algorithm:

- 
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
 2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
 3. update $\hat{Q}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{Q}_\phi^\pi(\mathbf{s}'_i, \mathbf{a}'_i)$ for each $\mathbf{s}_i, \mathbf{a}_i$
 4. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = Q(\mathbf{s}_i, \mathbf{a}_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
 5. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
 6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

not the action π_θ would have taken!

use the same trick, but this time for $\mathbf{a}_i$ rather than $\mathbf{a}'_i$!

sample $\mathbf{a}_i^\pi \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i^\pi|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i^\pi)$$

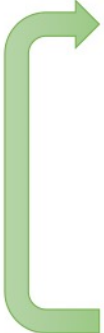
↑
not from replay buffer $\mathcal{R}$!

higher variance, but convenient
why is higher variance OK here?

in practice: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i^\pi|\mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}_i^\pi)$

What else is left?

online actor-critic algorithm:

- 
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
 2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
 3. update $\hat{Q}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{Q}_\phi^\pi(\mathbf{s}'_i, \mathbf{a}'_i)$ for each $\mathbf{s}_i, \mathbf{a}_i$
 4. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i^\pi | \mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}_i^\pi)$ where $\mathbf{a}_i^\pi \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$
 5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Is there any remaining problem?

$\mathbf{s}_i$ didn't come from $p_\theta(\mathbf{s})$

nothing we can do here, just accept it

intuition: we want optimal policy on $p_\theta(\mathbf{s})$
but we get optimal policy on a *broader* distribution

Off-policy Learning & Importance Sampling

$$\theta^* = \arg \max_{\theta} J(\theta)$$

$$J(\theta) = E_{\tau \sim p_{\theta}(\tau)}[r(\tau)]$$

what if we don't have samples from $p_{\theta}(\tau)$?

(we have samples from some $\bar{p}(\tau)$ instead)

$$J(\theta) = E_{\tau \sim \bar{p}(\tau)} \left[\frac{p_{\theta}(\tau)}{\bar{p}(\tau)} r(\tau) \right]$$

$$p_{\theta}(\tau) = p(s_1) \prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

$$\frac{p_{\theta}(\tau)}{\bar{p}(\tau)} = \frac{\cancel{p(s_1)} \prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \cancel{p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}}{\cancel{p(s_1)} \prod_{t=1}^T \bar{\pi}(\mathbf{a}_t | \mathbf{s}_t) \cancel{p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}} = \frac{\prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)}{\prod_{t=1}^T \bar{\pi}(\mathbf{a}_t | \mathbf{s}_t)}$$

importance sampling

$$\begin{aligned} E_{x \sim p(x)}[f(x)] &= \int p(x) f(x) dx \\ &= \int \frac{q(x)}{q(x)} p(x) f(x) dx \\ &= \int q(x) \frac{p(x)}{q(x)} f(x) dx \\ &= E_{x \sim q(x)} \left[\frac{p(x)}{q(x)} f(x) \right] \end{aligned}$$

Off-policy Policy Gradient

$$\theta^* = \arg \max_{\theta} J(\theta) \quad J(\theta) = E_{\tau \sim p_{\theta}(\tau)} [r(\tau)]$$

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim p_{\theta}(\tau)} \left[\frac{p_{\theta'}(\tau)}{p_{\theta}(\tau)} \nabla_{\theta'} \log \pi_{\theta'}(\tau) r(\tau) \right] \quad \text{when } \theta \neq \theta'$$

$$\frac{p_{\theta'}(\tau)}{p_{\theta}(\tau)} = \frac{\prod_{t=1}^T \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)}{\prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)}$$

$$= E_{\tau \sim p_{\theta}(\tau)} \left[\left(\prod_{t=1}^T \frac{\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)} \right) \left(\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right] \text{ what about causality?}$$

$$= E_{\tau \sim p_{\theta}(\tau)} \left[\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \underbrace{\left(\prod_{t'=1}^t \frac{\pi_{\theta'}(\mathbf{a}_{t'} | \mathbf{s}_{t'})}{\pi_{\theta}(\mathbf{a}_{t'} | \mathbf{s}_{t'})} \right)}_{\text{future actions don't affect current weight}} \left(\sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \right) \left(\prod_{t''=t}^{t'} \frac{\pi_{\theta'}(\mathbf{a}_{t''} | \mathbf{s}_{t''})}{\pi_{\theta}(\mathbf{a}_{t''} | \mathbf{s}_{t''})} \right) \right]$$

future actions don't affect current weight

if we ignore this, we get
a policy iteration algorithm
(more on this in a later lecture)

First-order Approximation of Importance Sampling

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim p_{\theta}(\tau)} \left[\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \underbrace{\left(\prod_{t'=t}^T \frac{\pi_{\theta'}(\mathbf{a}_{t'} | \mathbf{s}_{t'})}{\pi_{\theta}(\mathbf{a}_{t'} | \mathbf{s}_{t'})} \right)}_{\text{exponential in } T} \left(\sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \right) \right]$$

let's write the objective a bit differently...

on-policy policy gradient: $\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t}$

$(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \sim \pi_{\theta}(\mathbf{s}_t, \mathbf{a}_t)$

off-policy policy gradient: $\nabla_{\theta'} J(\theta') \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \frac{\pi_{\theta'}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})}{\pi_{\theta}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t}$

**We'll see why this is
reasonable
later in the course!**

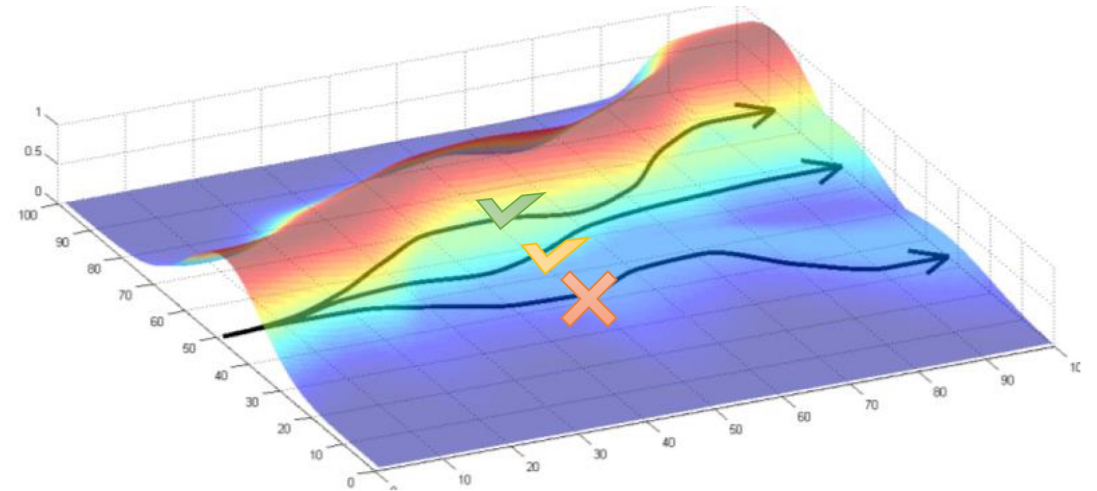
$$= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \frac{\cancel{\pi_{\theta'}(\mathbf{s}_{i,t})}}{\cancel{\pi_{\theta}(\mathbf{s}_{i,t})}} \frac{\pi_{\theta'}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t})}{\pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t}$$

ignore this part

Problem of Importance Sampling

- Recall that we ignore the state distribution mismatch between π_θ and $\pi_{\theta_{old}}$.
- If directly optimize $J(\theta)$, $\frac{\pi_\theta}{\pi_{old}}$ will become larger and larger, and thus cause significant distribution change.
 - Distribution change may lead policy network performance much worse
 - Policy sample bad actions, lead to worse policy.
 - Training collapsed!

$$J(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} \hat{A}_t \right]$$



TRPO

- TRPO: Optimize mean reward on a constraint (use importance sampling)

$$\theta_{k+1} = \arg \max_{\theta} \mathcal{L}(\theta_k, \theta) \quad \longleftarrow \quad \mathcal{L}(\theta_k, \theta) = \mathbb{E}_{s, a \sim \pi_{\theta_k}} \left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a) \right]$$

s.t. $\bar{D}_{KL}(\theta || \theta_k) \leq \delta$

- Intuition: π_{θ} shouldn't change too much within a single step.
 - Equals to change the step length in every step, avoiding training collapse.
 - It can be proved that the return is monotonous.
- TRPO uses hard constraints to control the change.
 - monotonous -> local minima
 - complicate math

TRPO

- How to optimize

$$\begin{aligned}\theta_{k+1} &= \arg \max_{\theta} \mathcal{L}(\theta_k, \theta) \\ \text{s.t. } \bar{D}_{KL}(\theta || \theta_k) &\leq \delta\end{aligned}$$

- Step 1: Taylor expand the objective and constraint to leading order around θ_k .

$$\begin{aligned}\theta_{k+1} &= \arg \max_{\theta} g^T(\theta - \theta_k) \\ \text{s.t. } \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) &\leq \delta.\end{aligned}$$

$$\begin{aligned}g_k &= \nabla_{\theta} L(\theta_k) \\ \hat{H}_k &= \nabla_{\theta}^2 \left(\frac{1}{N} \sum_{i=1}^N \text{KL}[\pi_{\theta_k}(\cdot | s_i) || \pi_{\theta}(\cdot | s_i)] \right) \Big|_{\theta=\theta_k}\end{aligned}$$

- Step 2: Use Lagrangian duality and yield the solution.

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g.$$

TRPO

- Step 3: Due to the approximation errors introduced by the Taylor expansion, KL constraint may not be satisfied.
- Use backtracking line search. α^j is the smallest nonnegative integer such that $\pi_{\theta_{k+1}}$ satisfied KL constraint.

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g$$

- Step 4: Solve $H^{-1}g$ is very expensive.
- Use conjugate gradient to solve $Hx = g$, and use x as $H^{-1}g$.

TRPO – Algorithm

Algorithm 1 Trust Region Policy Optimization

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: Hyperparameters: KL-divergence limit δ , backtracking coefficient α , maximum number of backtracking steps K
- 3: **for** $k = 0, 1, 2, \dots$ **do**
- 4: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 5: Compute rewards-to-go $\hat{R}_t$.
- 6: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
- 7: Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) |_{\theta_k} \hat{A}_t.$$

- 8: Use the conjugate gradient algorithm to compute

$$\hat{x}_k \approx \hat{H}_k^{-1} \hat{g}_k,$$

where $\hat{H}_k$ is the Hessian of the sample average KL-divergence.

- 9: Update the policy by backtracking line search with

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{\hat{x}_k^T \hat{H}_k \hat{x}_k}} \hat{x}_k,$$

where $j \in \{0, 1, 2, \dots, K\}$ is the smallest value which improves the sample loss and satisfies the sample KL-divergence constraint.

- 10: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T (V_{\phi}(s_t) - \hat{R}_t)^2,$$

typically via some gradient descent algorithm.

- 11: **end for**

We will talk about it later
in PPO

Backtracking line search

Value update

PPO

- Following TRPO's intuition, PPO provides two methods to constraint policy distribution's change using the first-order algorithm.
 - Clipped surrogate objective
 - Adaptive KL penalty coefficient
- Soft constraint
 - Trade-off: reward monotonous and policy randomness

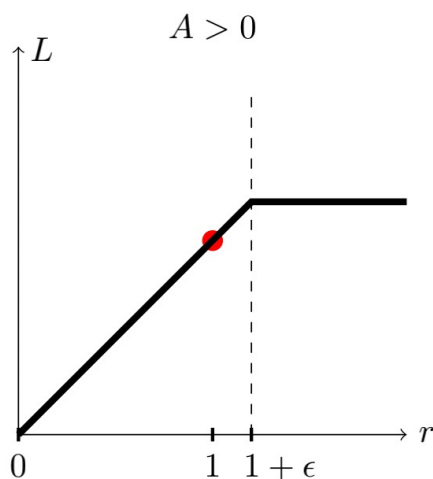
PPO – Clipped Surrogate Objective

- Clip the probability ratio to range $[1 - \epsilon, 1 + \epsilon]$

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{s, a \sim \pi_{\theta_k}} [L(s, a, \theta_k, \theta)], \quad L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \text{clip} \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right)$$

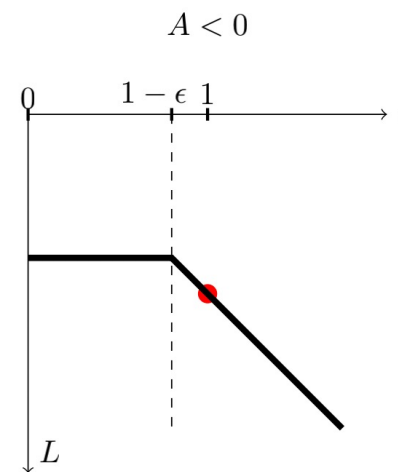
- Condition 1: $A^{\pi_{\theta_k}}$ is positive.

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}, (1 + \epsilon) \right) A^{\pi_{\theta_k}}(s, a)$$



- Condition 2: $A^{\pi_{\theta_k}}$ is negative.

$$L(s, a, \theta_k, \theta) = \max \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}, (1 - \epsilon) \right) A^{\pi_{\theta_k}}(s, a)$$



PPO – Adaptive KL Penalty Coefficient

- KL constraint in TRPO is hard to apply.
 - Adaptive KL penalty coefficient has similar effect.
- Adaptive KL can be alternative to clipped surrogate objective.
- Achieve some target value of KL divergence d_{target} .

$$L^{KL\text{PEN}}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t - \beta \text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_\theta(\cdot | s_t)] \right]$$

Compute $d = \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_\theta(\cdot | s_t)]]$

- If $d < d_{\text{target}}/1.5$, $\beta \leftarrow \beta/2$
- If $d > d_{\text{target}} \times 1.5$, $\beta \leftarrow \beta \times 2$

PPO – Generalized Advantage Estimation (GAE)

- Key idea: balances bias and variance in advantage estimation by combining multiple truncated advantage estimates.
 - $\hat{A}_t^{(1)} = r_t + \gamma V(s_{t+1}) - V(s_t) \rightarrow$ High bias, low variance.
 - $\hat{A}_t^{(2)} = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t) \rightarrow$ Moderate bias/variance.
 - $\hat{A}_t^{(\infty)} = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t) \rightarrow$ Unbiased, high variance.
- Combines these estimates with weights with $\omega_k = \lambda^{k-1}$

$$\hat{A}_t^{\text{GAE}} = \frac{\sum_k w_k \hat{A}_t^{(k)}}{\sum_k w_k} \quad \text{equals to} \quad \hat{A}_t^{\text{GAE}} = \delta_t + \gamma \lambda \delta_{t+1} + \cdots + (\gamma \lambda)^{T-t+1} \delta_{T-1}$$
$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

PPO Algorithm

Algorithm 1 PPO-Clip

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 4: Compute rewards-to-go $\hat{R}_t$.
- 5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
- 6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

- 7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

- 8: **end for**

Usually use GAE here

$$g(\epsilon, A) = \begin{cases} (1 + \epsilon)A & A \geq 0 \\ (1 - \epsilon)A & A < 0 \end{cases}$$

Value update

Implementation Tricks of PPO

- <https://zhuanlan.zhihu.com/p/512327050>
- <https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details>
- Engstrom, Logan, et al. "Implementation matters in deep policy gradients: A case study on PPO and TRPO." arXiv preprint arXiv:2005.12729. ²⁷

PPO – Update tricks

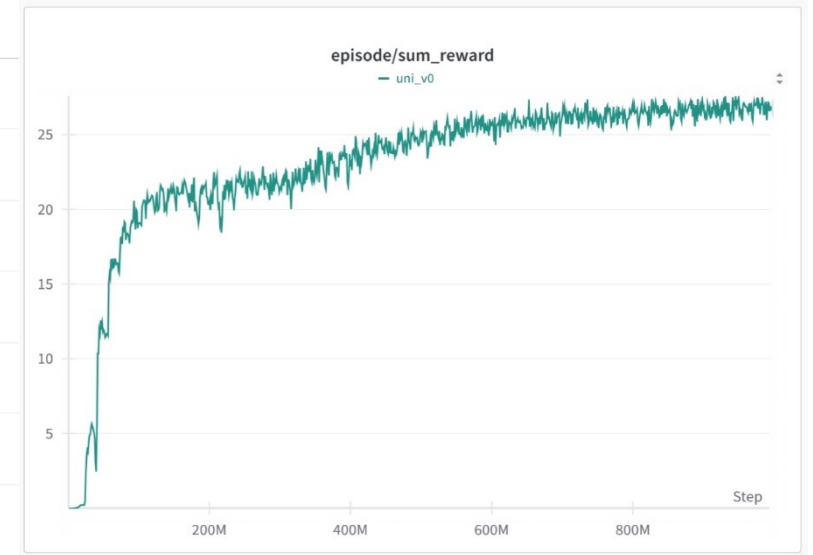
- 
- 1, sample action-state pairs $\{s_t, a_t, s_{t+1}\} \rightarrow B$ from π_θ
 - 2, update $V_\phi, \pi_{\theta'}$ **for K epochs**, each epoch use all of data in B
 - 3, $\theta' \rightarrow \theta$

- In practice, K shouldn't be too large, otherwise the distribution mismatch would be large.
- Mini-batch: In each epoch, we divide buffer into several mini-batches (similar to batch gradient decent).
 - Too large mini-batch: stuck in local minima
 - Too small mini-batch: noisy gradient

PPO – Hyperparameter

- Hyperparameter used in quadrupedal robot locomotion task
- Comparison between different implementations and hyperparameters

Hyperparameter	Value
Discount Factor (γ)	0.98
Clipping ϵ	0.2
GAE λ	0.95
Learning Rate	0.0003
Number of Envs	16,384
Mini-batch Size	512
Update Iterations (K)	5



algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	0.82
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

PPO – Normalization / Scaling

- Advantage normalization: $A_t^{norm} = \frac{A_t - \mu_A}{\sqrt{\sigma_A^2 + \epsilon}}$
 - Only used in training time
 - Normalize in a minibatch
 - When the value function is not well trained, it helps A_t center around 0.
- State normalization : $s_t^{norm} = \frac{s_t - \mu_{run}}{\sqrt{\sigma_{run}^2 + \epsilon}}$
 - Need to restore the running mean of states and use it in test time.
 - Similar to Batch-Norm, it helps with training.
- Reward scaling : $r_t^{scale} = \frac{r_t}{\sqrt{\sigma_{run}^2 + \epsilon}}$
 - Scaled reward makes value function $V_\phi(s_t)$ training easier.

PPO – Initialization & Activation

- Orthogonal Initialization
 - Step 1: Gaussian distribution($\mu = 0$, $\sigma = 1$) initialize W .
 - Step 2: Apply SVD, $W = U^T D V$, $V \rightarrow W$. W is an orthogonal matrix.
 - [Engstrom, Ilyas, et al., \(2020\)](#) find orthogonal initialization to outperform the default Xavier initialization.
 - [Andrychowicz, et al. \(2021\)](#) find centering the action distribution around 0 (i.e., initialize the policy output layer weights with 0.01”) to be beneficial
- Tanh activation
 - use Tanh activation instead of ReLU (PPO original paper)
 - only 2 or 3 hidden layer MLP, gradient vanishing may not have significant damage.

Pros and Cons

- Pros:
 - Better sample efficiency
 - Importance sampling, minibatch
 - More stable training and better performance
 - Clipped surrogate objective, adaptive KL, other implemental tricks
 - Simpler math than TRPO
- Cons:
 - Still poor sample efficiency compared with off-policy RL algorithm
 - Need to train in simulator, has domain gap to real-world environment
 - Many task-specific tricks
 - Carefully choose hyperparameters

Further Reading and Practice

OpenAI Tutorial to RL

<https://spinningup.openai.com/en/latest/user/introduction.html>



OpenAI
Spinning Up

ALGORITHMS DOCS

Vanilla Policy Gradient

Trust Region Policy Optimization

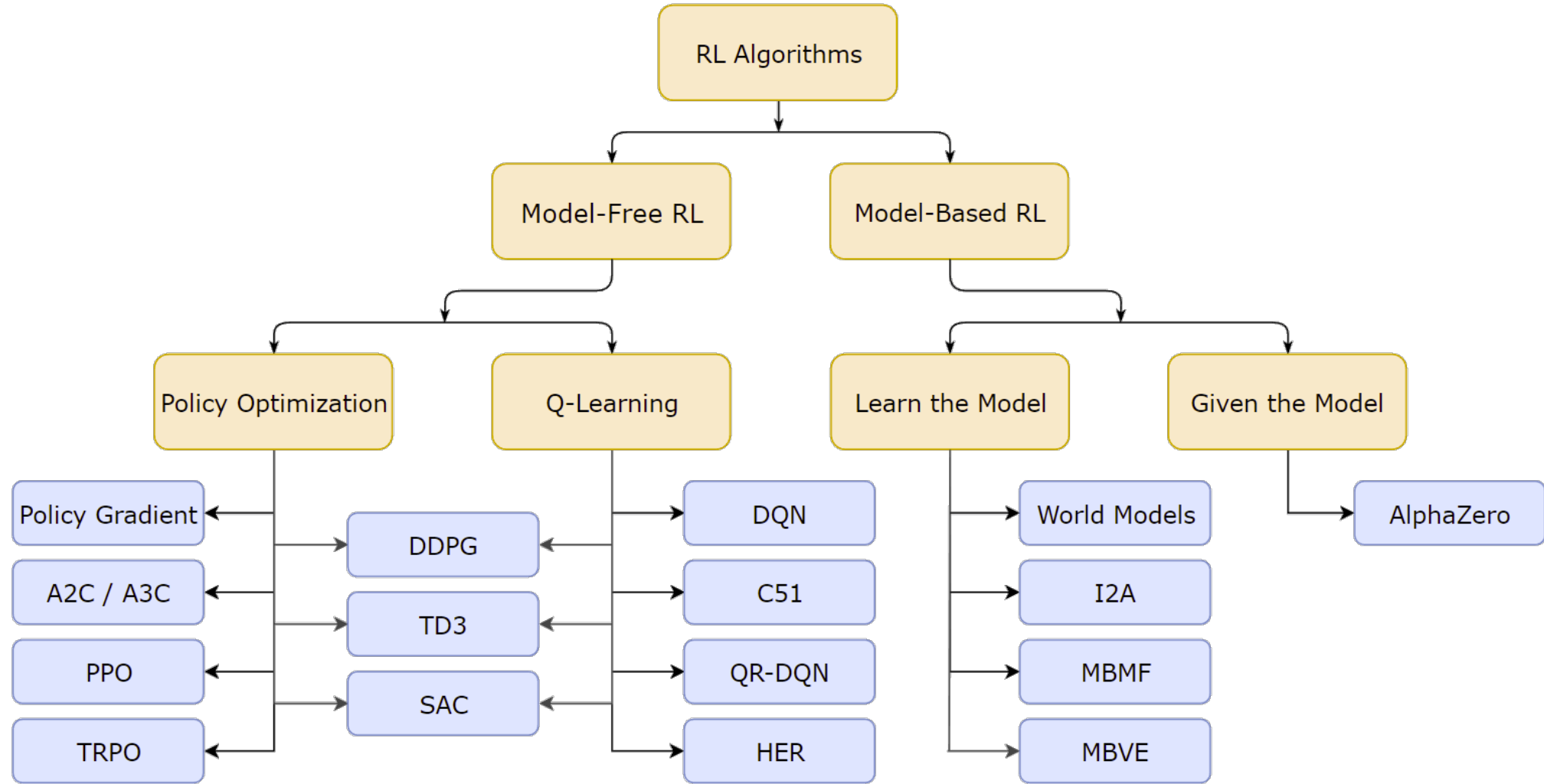
Proximal Policy Optimization

Deep Deterministic Policy Gradient

Twin Delayed DDPG

Soft Actor-Critic

A Taxonomy of RL Algorithms



Locomotion

Recent Advances

Unitree B2-W

- Terrain traverse
- High-dynamic movements



Tesla Optimus

- Terrain traverse
- Robustness



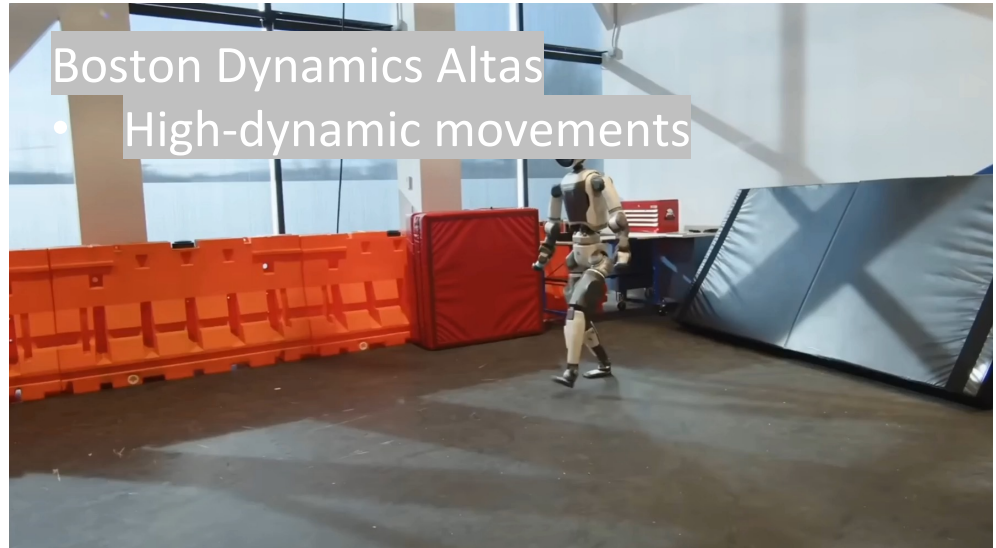
Unitree G1

- Robustness



Boston Dynamics Atlas

- High-dynamic movements



Legged Robot



**Humanoid robot from
Boston Dynamics**

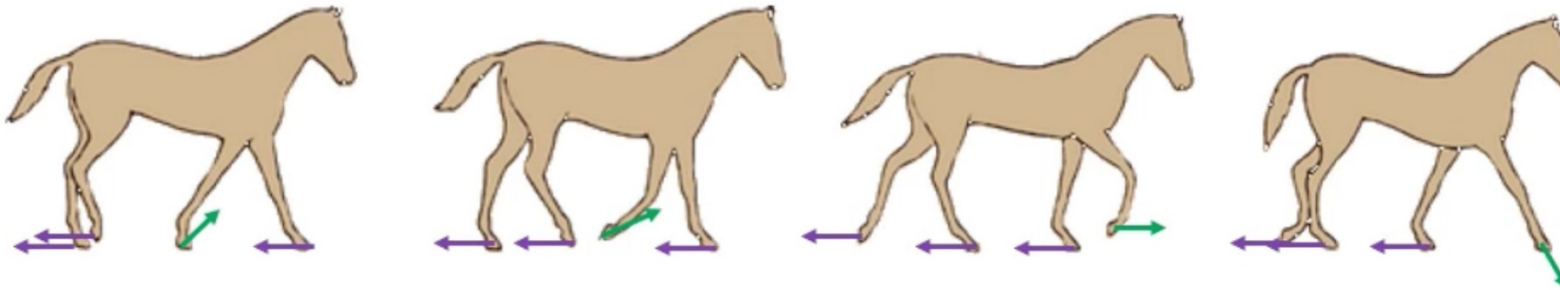


**quadrupedal robot
from Unitree**

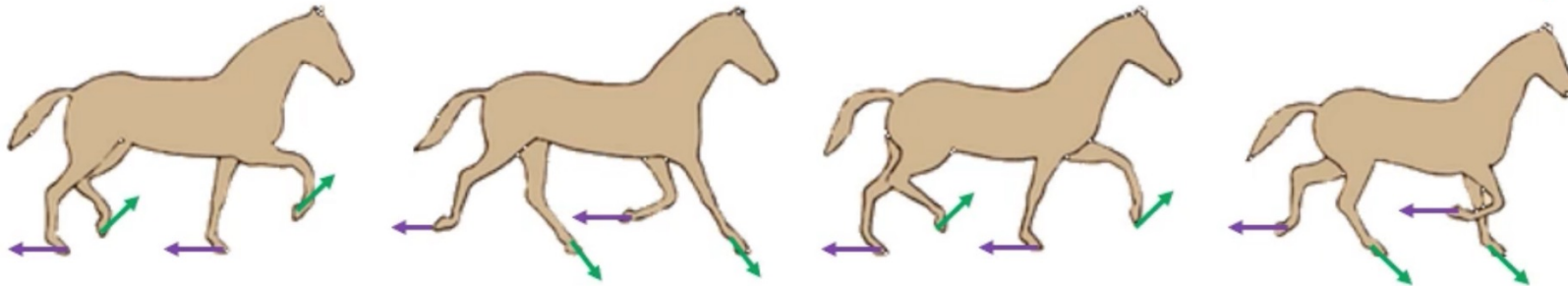
Gait

Gait: the pattern of how does a person/animal walk

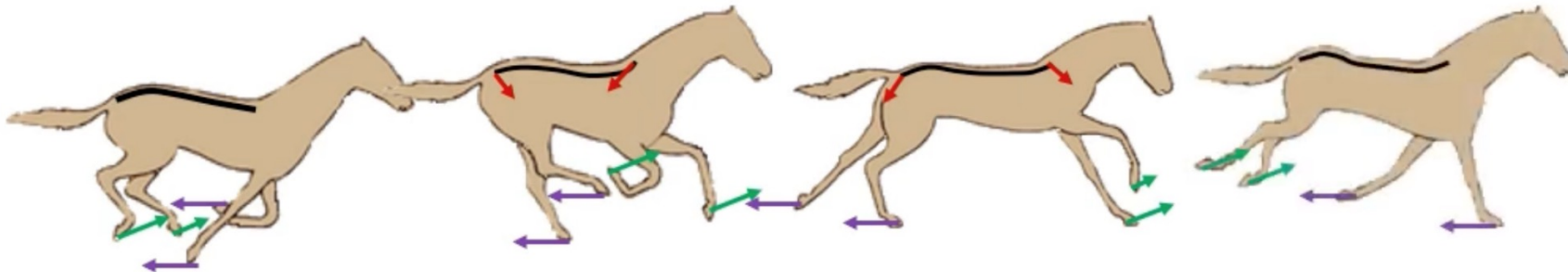
crawl
(walk)



trot



gallop



Popular Approaches

- Traditional model-based methods
 - Model predictive control (MPC)
 - Trajectory optimization
- Reinforcement learning
 - Model-free RL
 - Faster than MPC
 - More flexible than MPC
 - Model-based RL

Model Predictive Control (MPC)

- Definition: At each time step t_s , solve the following optimization problem and apply $u(t_s)$ as action.

$$\begin{array}{llll} \underset{\mathbf{u}(\cdot)}{\text{minimize}} & \int_{t_s}^{t_f} l(\mathbf{x}(t), \mathbf{u}(t), t) dt, & (1a) & \text{Cost} \\ \text{subject to} & \mathbf{x}(t_s) = \mathbf{x}_s, & (1b) & \text{Initial condition} \\ & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), & (1c) & \text{System dynamic} \\ & \mathbf{g}(\mathbf{x}, \mathbf{u}, t) = \mathbf{0} & (1d) & \text{Equality constraints} \\ & \mathbf{h}(\mathbf{x}, \mathbf{u}, t) \geq \mathbf{0}, & (1e) & \text{Inequality Constraints} \end{array}$$

where $x(t)$ is the state and $u(t)$ is the control input.

An Example: Inverted Pendulum

Control input

$$\mathbf{u}(t) = [F(t)]$$

State

$$\mathbf{x}(t) = \begin{bmatrix} y(t) \\ \theta(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix}$$

• Cost

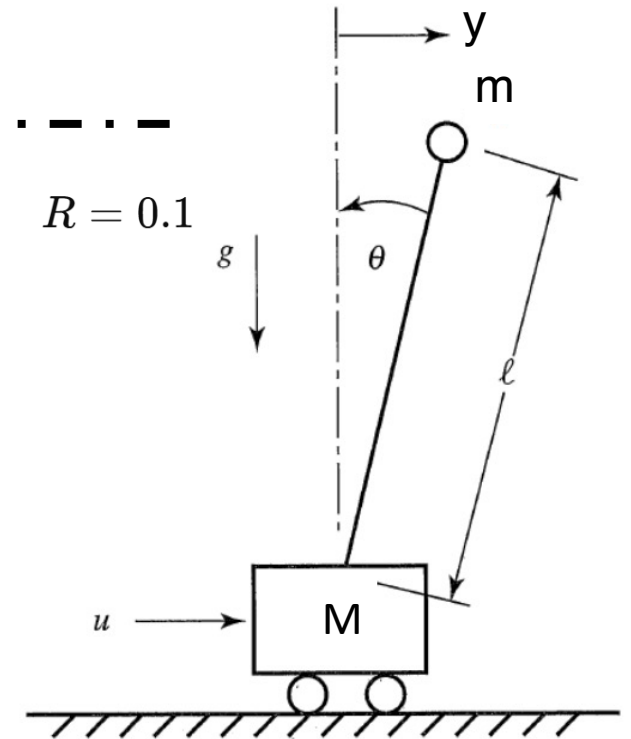
$$\min_{\mathbf{u}(\cdot)} \int_{t_s}^{t_f} [\mathbf{x}(t)^T Q \mathbf{x}(t) + \mathbf{u}(t)^T R \mathbf{u}(t)] dt \quad Q = \text{diag}(10, 100, 1, 10), \quad R = 0.1$$

• System dynamic

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{y} \\ \dot{\theta} \\ \frac{F + ml\dot{\theta}^2 \sin \theta - mg \sin \theta \cos \theta}{M + m \sin^2 \theta} \\ \frac{-F \cos \theta - ml\dot{\theta}^2 \sin \theta \cos \theta + (M + m)g \sin \theta}{l(M + m \sin^2 \theta)} \end{bmatrix}$$

• Constraints

$$\mathbf{h}(\mathbf{x}, \mathbf{u}, t) = \begin{bmatrix} y(t) + y_{\max} \\ y_{\max} - y(t) \\ F(t) + F_{\max} \\ F_{\max} - F(t) \end{bmatrix} \geq 0$$



An Example: Match Man

Control input

$$\mathbf{u}(t) = \begin{bmatrix} \tau_L(t) \\ \tau_R(t) \end{bmatrix}$$

State

$$\mathbf{x}(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \\ \theta_1(t) \\ \dot{\theta}_1(t) \\ \theta_2(t) \\ \dot{\theta}_2(t) \end{bmatrix}$$

- Cost

$$\min_{\mathbf{u}(\cdot)} \int_{t_s}^{t_f} \left[w_1 (\dot{y}(t) - v_{\text{target}})^2 + w_2 \|\mathbf{u}(t)\|^2 \right] dt$$

- System dynamic

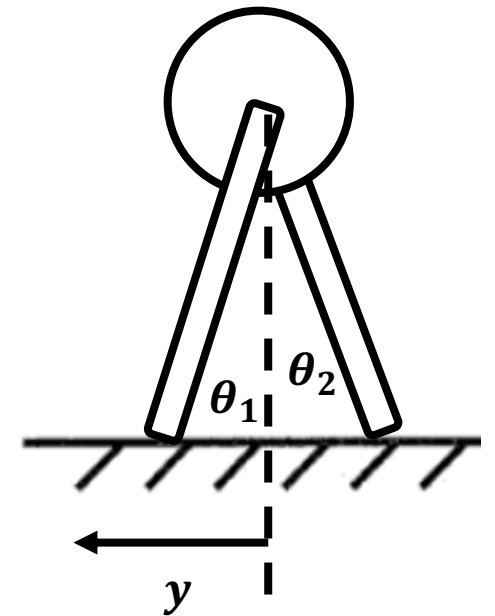
$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} \dot{y}(t) \\ \ddot{y}(t) \\ \dot{\theta}_1(t) \\ \ddot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ \ddot{\theta}_2(t) \end{bmatrix}$$

Coriolis and centrifugal forces

inertia matrix gravitational forces actuation matrix

- Constraints

$$|\tau_L(t)| \leq \tau_{\max}, \quad |\tau_R(t)| \leq \tau_{\max}$$



Contact constraint and dynamics are extremely hard to model!

Examples of MPC

ASIMO from Honda



Atlas from Boston Dynamics



Upright walking and basic movements, controlled by MPC + WBC

Demo: Boston Dynamics



Examples of RL

宇树 Unitree H1



天工 Tien Kung



Complex-terrain walking, running, and human-like dancing, powered by end-to-end reinforcement learning based on proprioceptive inputs

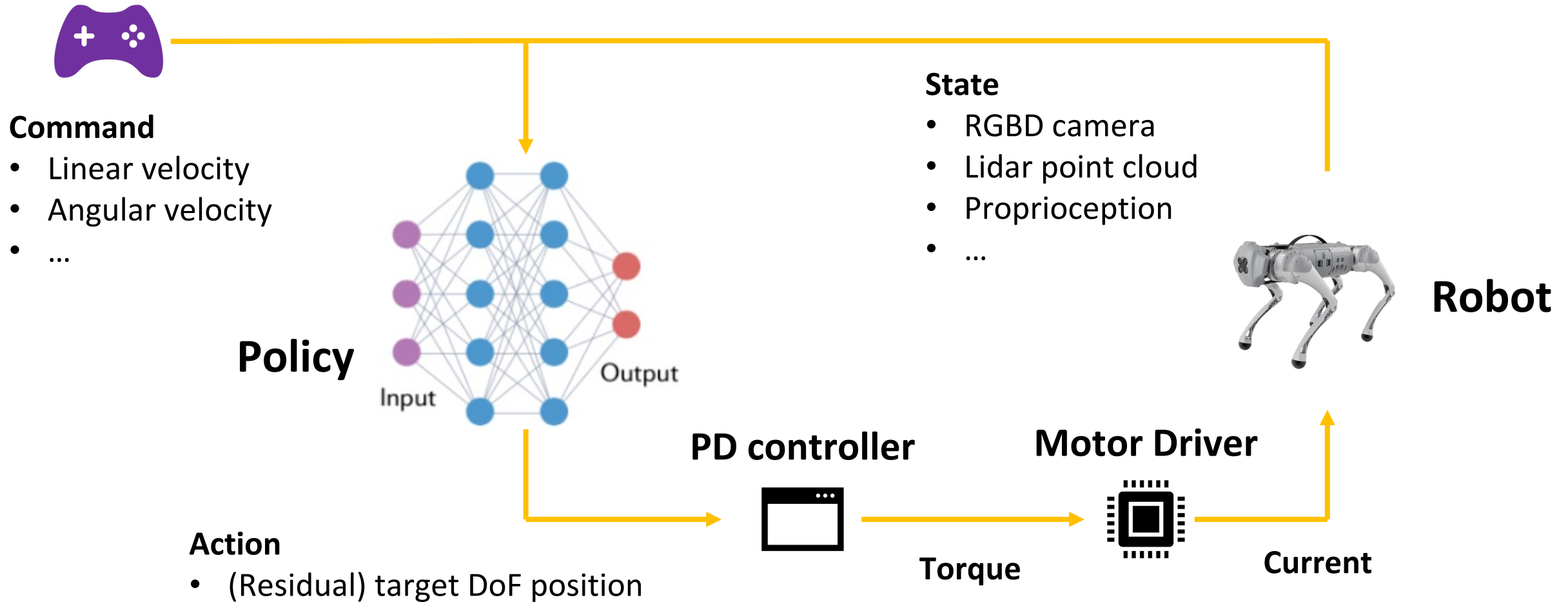
Why Use RL in Locomotion

- Extremely complex dynamics
 - Contacts and other constraints are hard to model.
 - MPC-based locomotion lacks robustness to disturbances.
- High difficulty in data acquisition
 - Strong coupling between data and configuration.
 - Teleoperation + imitation is a common practice in manipulation.
However, it's almost impossible to do teleoperation in locomotion.
- Naturally integrates perception and decision-making

MPC vs RL

Aspect	MPC	RL
Model Requirement	Yes	No
Constraint Handling	Naturally handles constraints	Handle constraints through reward
Real-Time Optimization	Solves optimization when deploying	Requires a learning stage
Environment Adaptability	Suitable for known and stable environments	Capable of adapting to complex and unknown environments
Handling High-Dimension	Optimization can be complex	Performs well with high-dimensional state and action spaces
Long-Term Decision Making	Optimizes a fixed prediction horizon	Naturally handles long-term decision

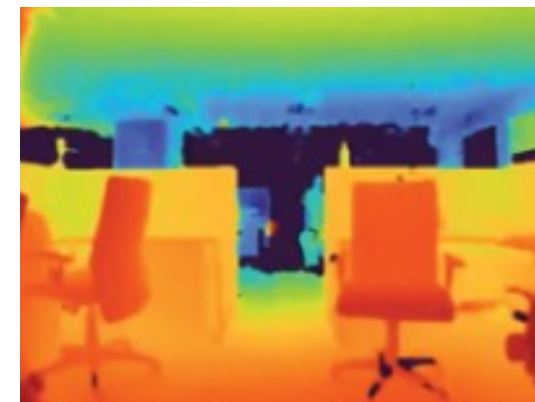
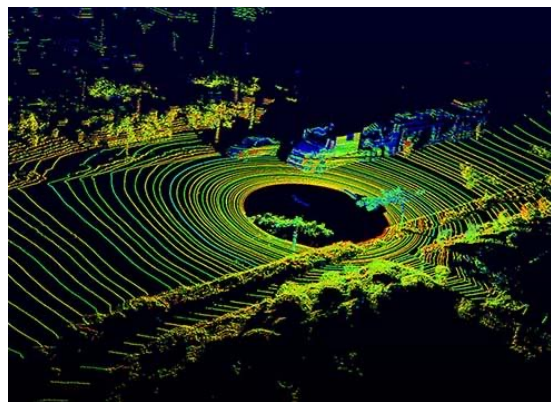
How Agent interacts with Robot



In the following we will take Unitree Go2 as an example

MDP Formulation (State)

- Proprioception
 - IMU (Inertial Measurement Unit)
 - Accelerometer a_x, a_y, a_z
 - Gyroscope $\omega_x, \omega_y, \omega_z$
 - Magnetometer (used to correct gyroscope drift)
 - Joint encoders
 - Joint position
 - Joint velocity
 - Contact sensors (optional)
- Exteroception (optional)
 - Lidar point cloud
 - RGBD camera
- Goal
 - Velocity (v_x, v_y, ω_z)



MDP Formulation (State)

- Two ways of representing body pose

- Joint space

- $[\theta_1, \dot{\theta}_1, \dots, \theta_n, \dot{\theta}_n]$ (12 joints)
- Inputs to the joint encoder
- Low-dimension, easy to train
- Align with action space

Used in locomotion tasks 

- Link space

- $[(p_1, q_1, v_1, \omega_1), \dots, (p_m, q_m, v_m, \omega_m)]$ (13 links)
- Encodes spatial relationships

Used in HOI, dancing tasks

MDP Formulation (Action)

- Position PD control
 - Usually, action will output residual DoF position

$$e(t) = \pi(s_t) + \theta_0 - \theta \quad \tau_t = K_p e(t) + K_d \frac{d}{dt} e(t)$$

- θ_0 are the reference positions

- Velocity PD control $\tau_t = K_p e_v(t) + K_d \frac{d}{dt} e_v(t)$

- $e_v(t) = \pi(s_t) - \dot{\theta}(t)$

- Torque control

- $\tau_t = \pi(s_t)$

MDP Formulation (Action)

- Position PD control is the most widely used in locomotion tasks
 - vs. torque control: PD policy can be updated very slowly (e.g. 50 Hz), but still output torque with high frequency (e.g. 1000 Hz).
 - vs. velocity PD control: Position PD control performs better in sim-to-real.
- Why PID is not used?
 - In locomotion and other non-stationary motions, the integral term continuously accumulates errors, leading to oscillations in the control signal.
 - RL inherently possesses compensation capabilities similar to the integral term.
 - Most importantly, RL+PD performs well.

MDP Formulation (Reward)

- Currently there's no general set of rules to define the components of the reward function.
- Tuning the coefficients of every reward term is extremely time-consuming.

MDP Formulation (Reward)

- What rewards do we need for training a quadruped robot to stand up?

- Task rewards

$$r_{\text{upright}} = \mathbf{z}_{\text{base}} \cdot [0, 0, 1]$$

$$r_{\text{height}} = \text{clip}(h_{\text{base}} - h_{\text{min}}, 0, r_{\text{max}})$$

$$r_{\text{contact}} = \frac{1}{4} \sum_{i=1}^4 1_{\text{contact}_i}$$

- Regularization rewards

$$r_{\text{joint_limit}} = - \sum_j \exp(\alpha \cdot (q_j - q_{j,\text{limit}})^2)$$

$$r_{\text{energy}} = - \sum_j^i |\tau_j \cdot \dot{q}_j|$$

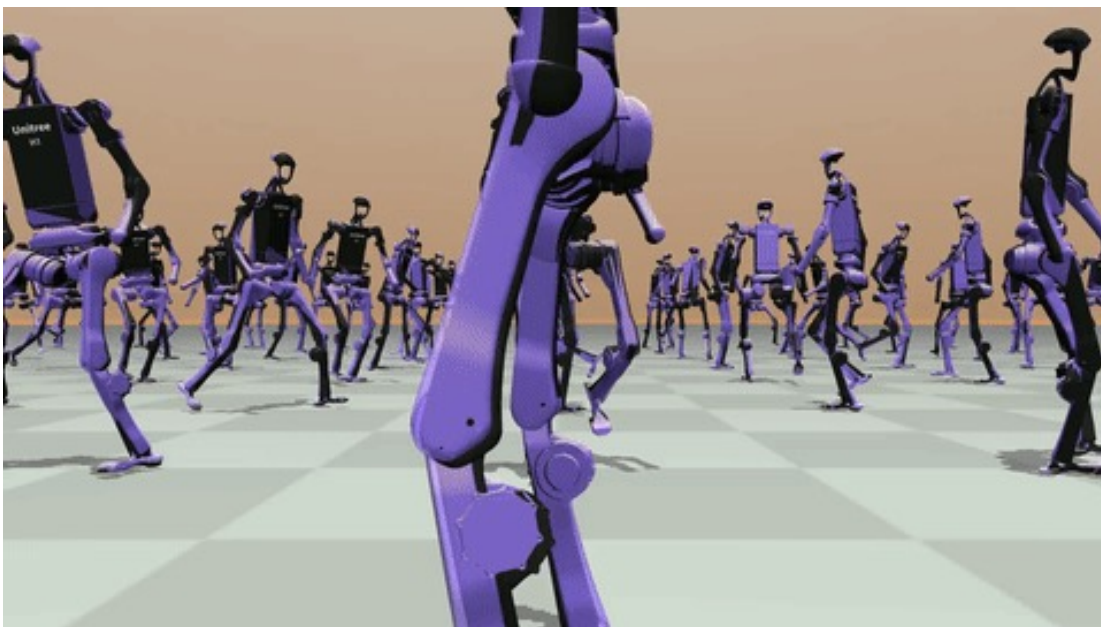
If the robot exhibits undesirable motions, additional regularization rewards is needed.

MDP Formulation (Reward)

Common rewards used in motion tracking

- Using reference related reward is a way to alleviate the heavy reward design.

$$\phi(x) := \exp(-k\|x\|^2)$$

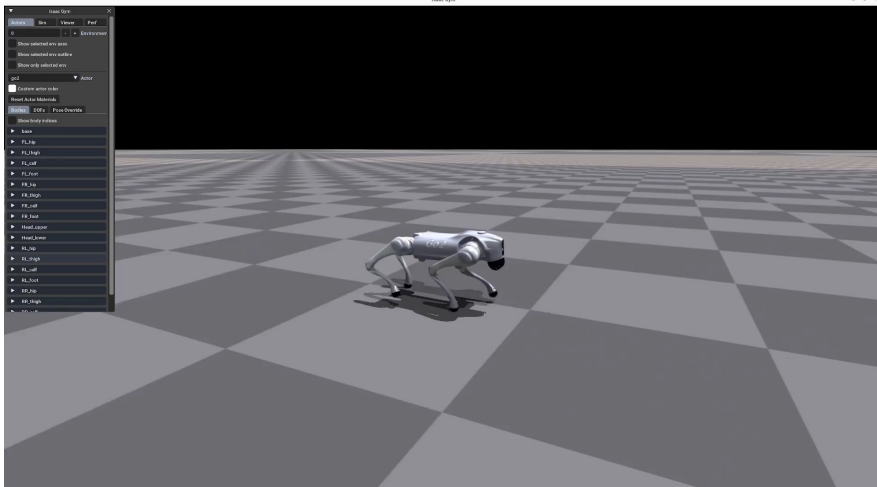


Horizontal velocity tracking	$\phi(v_{xy}^* - v_{xy})$
Yaw rate tracking	$\phi(\omega_z^* - \omega_z)$
Base z motion	$-v_z^2$ or $\phi(v_z)$
Roll and pitch rate orientation	$- \omega_{xy} ^2$ or $\phi(\omega_{xy})$
Joint velocities	$-\sum_{i \in \text{joints}} \dot{q}_i ^2$
Joint accelerations	$-\sum_{i \in \text{joints}} \ddot{q}_i ^2$
Joint torques	$-\sum_{i \in \text{joints}} \tau_i ^2$
Joint mechanical power	$-\sum_{i \in \text{joints}} (\tau_i * \dot{q}_i)^2$
Action rate	$- \mathbf{a}_t - \mathbf{a}_{t-1} ^2$
Action smoothness	$- \mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2} ^2$

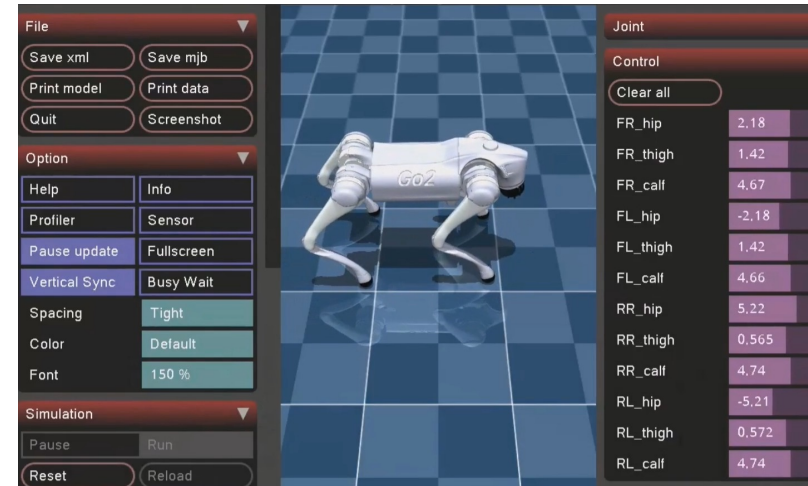
Refer to <https://github.com/robfirmas/loco-mujoco>

MDP Formulation (Transition/Simulator)

- **ISAAC** Series
- Issac Gym, Isaac Sim, Isaac Lab

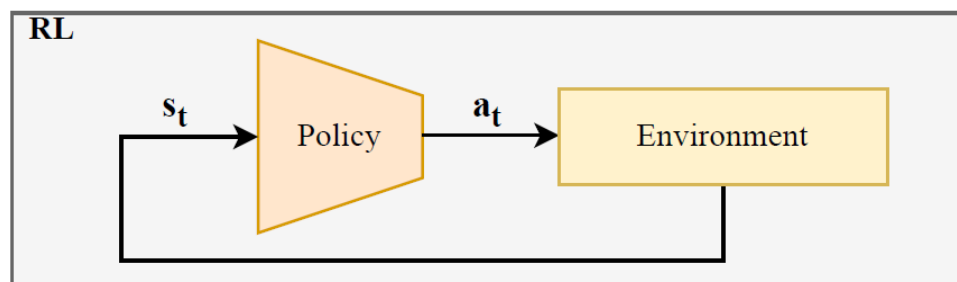


- **MuJoCo** Series
- MuJoCo, MuJoCo-Playground



Frameworks – End-to-End Learning

- Why robotics researcher always use PPO?
 - In highly dynamic tasks, the Q-function requires in SAC / DDPG / ... exhibits sharp variations with respect to changes in action.
 - With parallel simulation, sample-efficiency of on-policy methods is no-longer a problem.



Frameworks – Curriculum Learning

Example 1: Terrain traverse

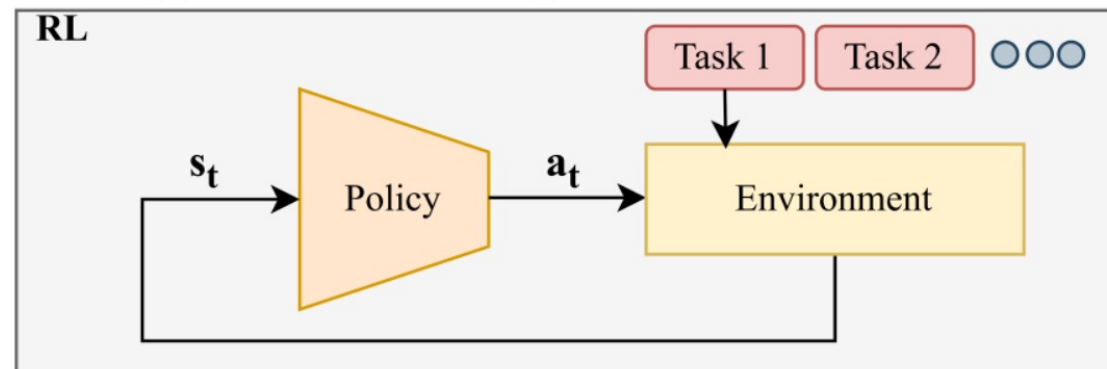
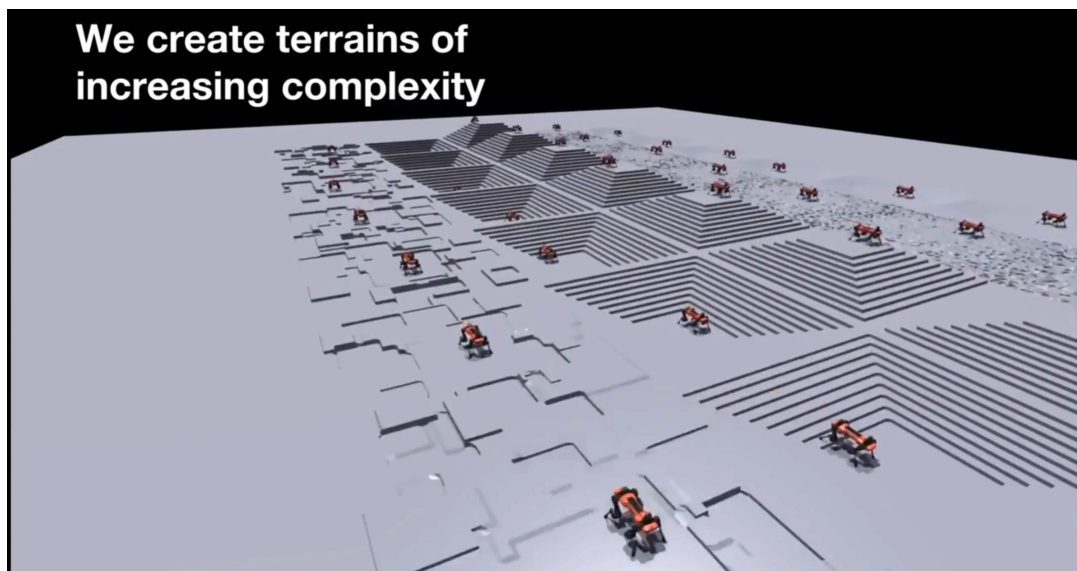
Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning

Nikita Rudin
ETH Zurich and NVIDIA
rudinn@ethz.ch

David Hoeller
ETH Zurich and NVIDIA
dhoeller@ethz.ch

Philipp Reist
NVIDIA
preist@nvidia.com

Marco Hutter
ETH Zurich
mahutter@ethz.com



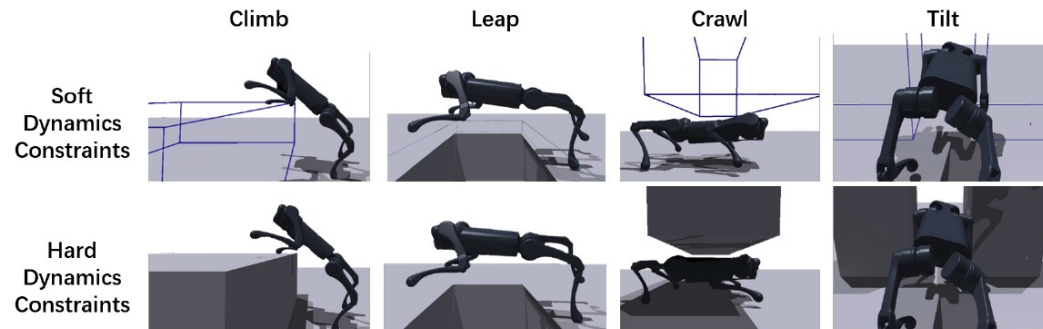
Frameworks – Curriculum Learning

Example 2: Obstacle avoidance

Robot Parkour Learning

Ziwen Zhuang^{*13} Zipeng Fu^{*2} Jianren Wang⁴ Christopher Atkeson⁴ Sören Schwertfeger³
Chelsea Finn² Hang Zhao¹⁵

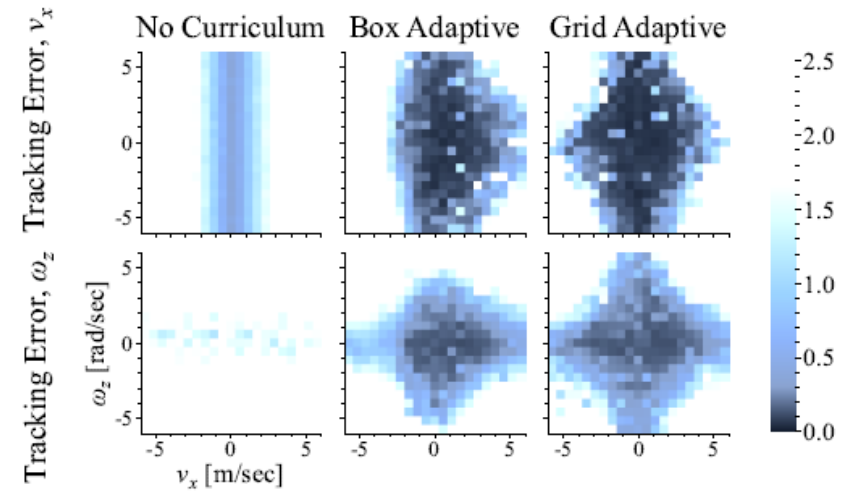
¹Shanghai Qi Zhi, ²Stanford, ³ShanghaiTech, ⁴CMU, ⁵Tsinghua, ^{*}project co-leads



Example 3: Fast moving

Rapid Locomotion via Reinforcement Learning

Gabriel B. Margolis^{*1}, Ge Yang^{*1 2}, Kartik Paigwar¹, Tao Chen¹, and Pulkit Agrawal^{1 2}
¹MIT Improbable AI Lab ²NSF AI Institute for Artificial Intelligence and Fundamental Interactions
Massachusetts Institute of Technology, Cambridge, MA 02139



Learning Frameworks – Hierarchical Learning

Example 2: Terrain traverse (vision-based)

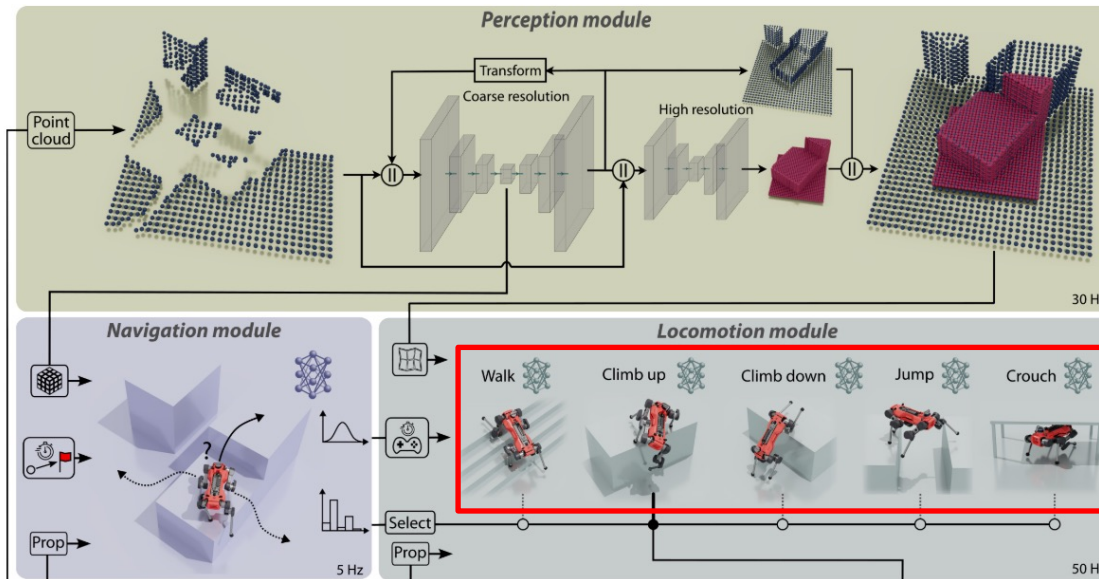
ANYmal Parkour: Learning Agile Navigation for Quadrupedal Robots

David Hoeller
ETH Zurich
NVIDIA
dhoeller@ethz.ch
Equal contribution

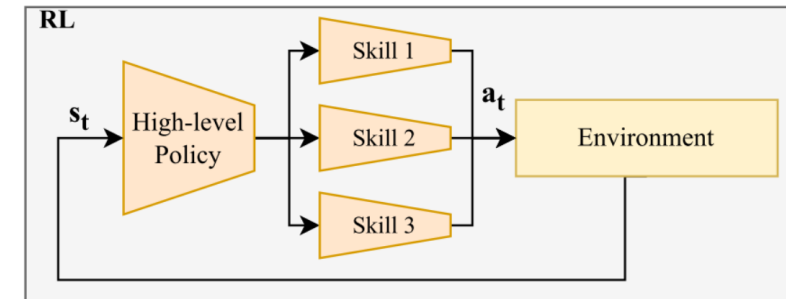
Nikita Rudin
ETH Zurich
NVIDIA
rudinn@ethz.ch
Equal contribution

Dhionis Sako
ETH Zurich
dsako@ethz.ch

Marco Hutter
ETH Zurich
mahutter@ethz.ch



Skill
Library



Learning Frameworks – Privileged Learning

- Why some privilege information (linear velocity, base height) so important?
 - Imaging a dog jumping over a gap
 - Actually, the dog does not know where it is with only proprioception

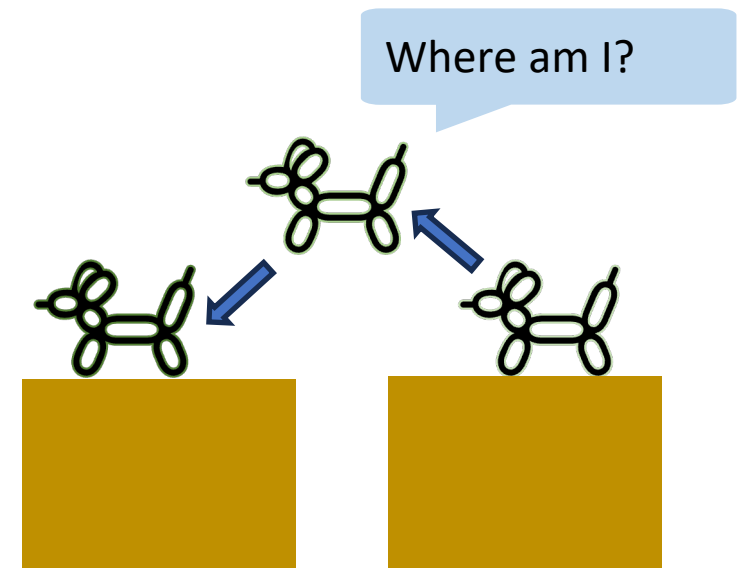
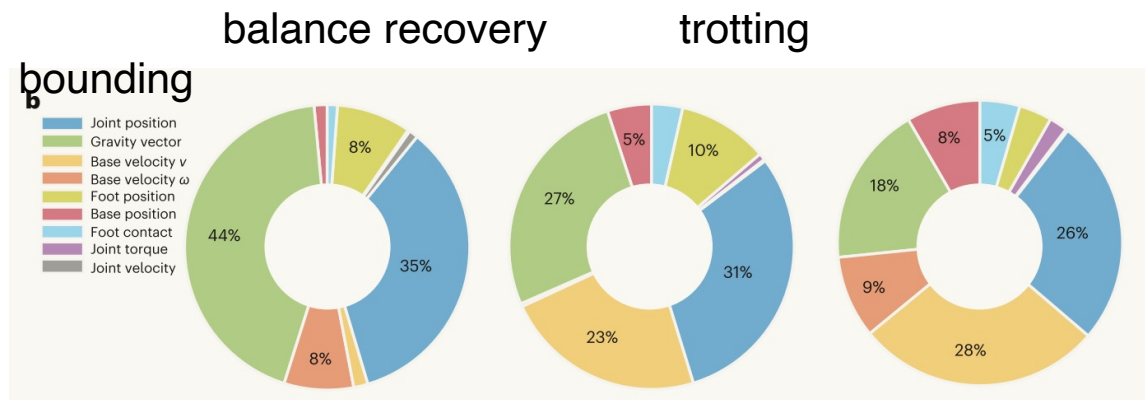
- Some analysis **Identifying important sensory feedback for learning locomotion skills**

Received: 3 December 2022

Accepted: 30 June 2023

Published online: 21 August 2023

Wanming Yu^{1,5}, Chuanyu Yang^{1,2,5}, Christopher McGreavy¹,
Eleftherios Triantafyllidis¹, Guillaume Bellegarda³, Milad Shafiee³,
Auke Jan Ijspeert³ & Zhibin Li⁴✉

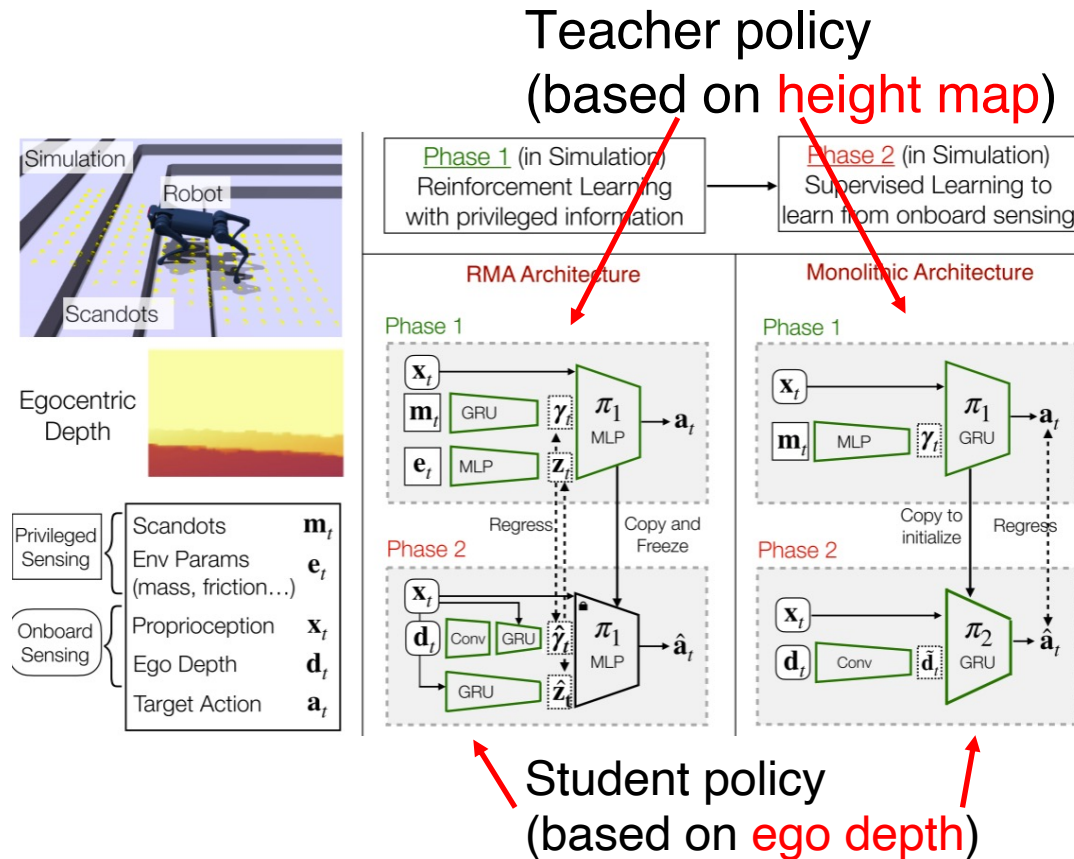


Learning Frameworks – Privileged Learning

Example 1: Policy Distill

Legged Locomotion in Challenging Terrains using Egocentric Vision

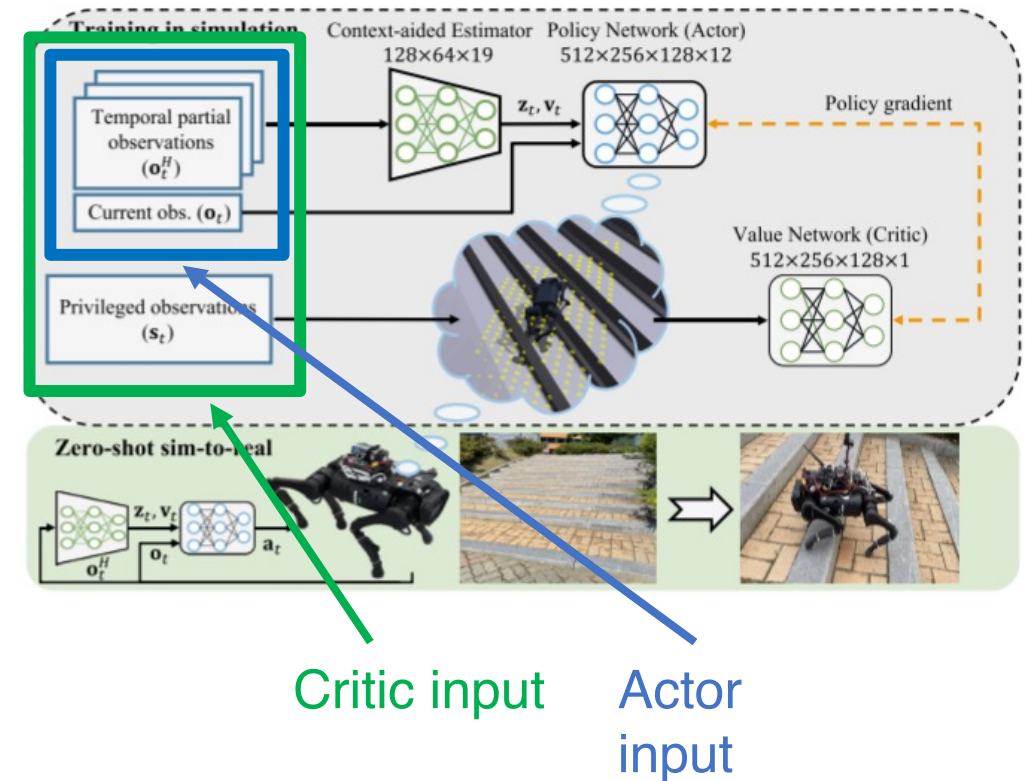
Ananye Agarwal^{*1}, Ashish Kumar^{*2}, Jitendra Malik^{†2}, Deepak Pathak^{†1}
¹Carnegie Mellon University, ²UC Berkeley



Example 2: Asymmetric Actor-Critic

DreamWaQ: Learning Robust Quadrupedal Locomotion With Implicit Terrain Imagination via Deep Reinforcement Learning

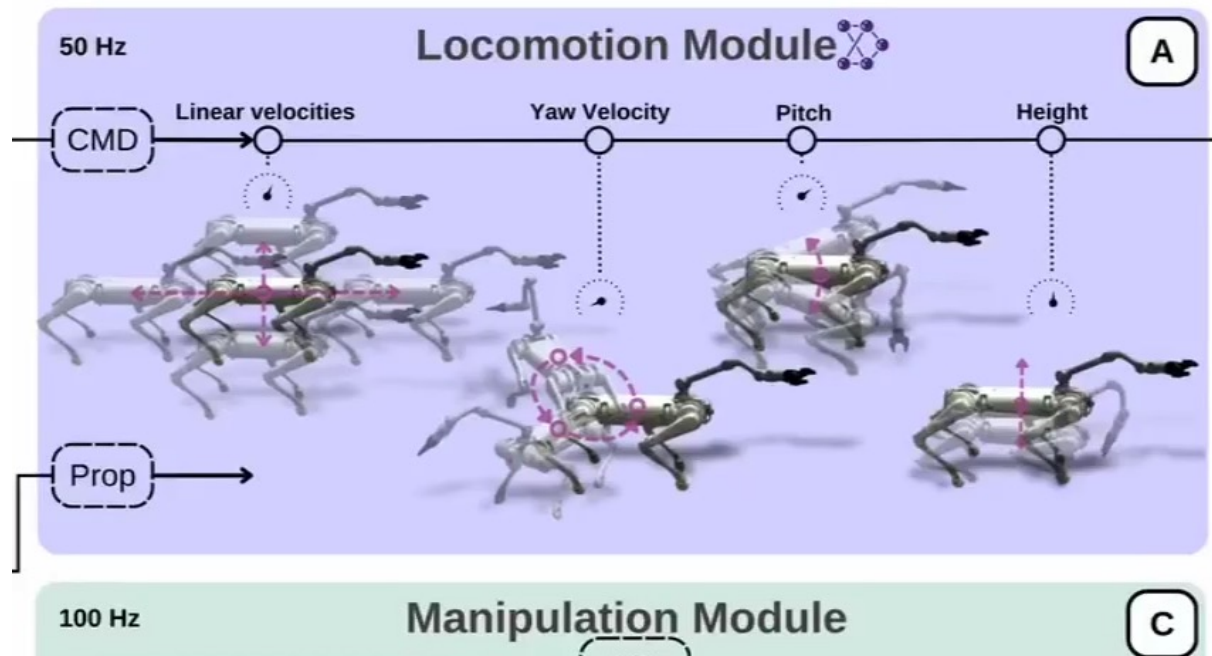
I Made Aswin Nahrendra¹, Byeongho Yu¹, and Hyun Myung^{1*}, *Senior Member, IEEE*



Example of Locomanipulation: QuadWBG



QuadWBG - Locomotion



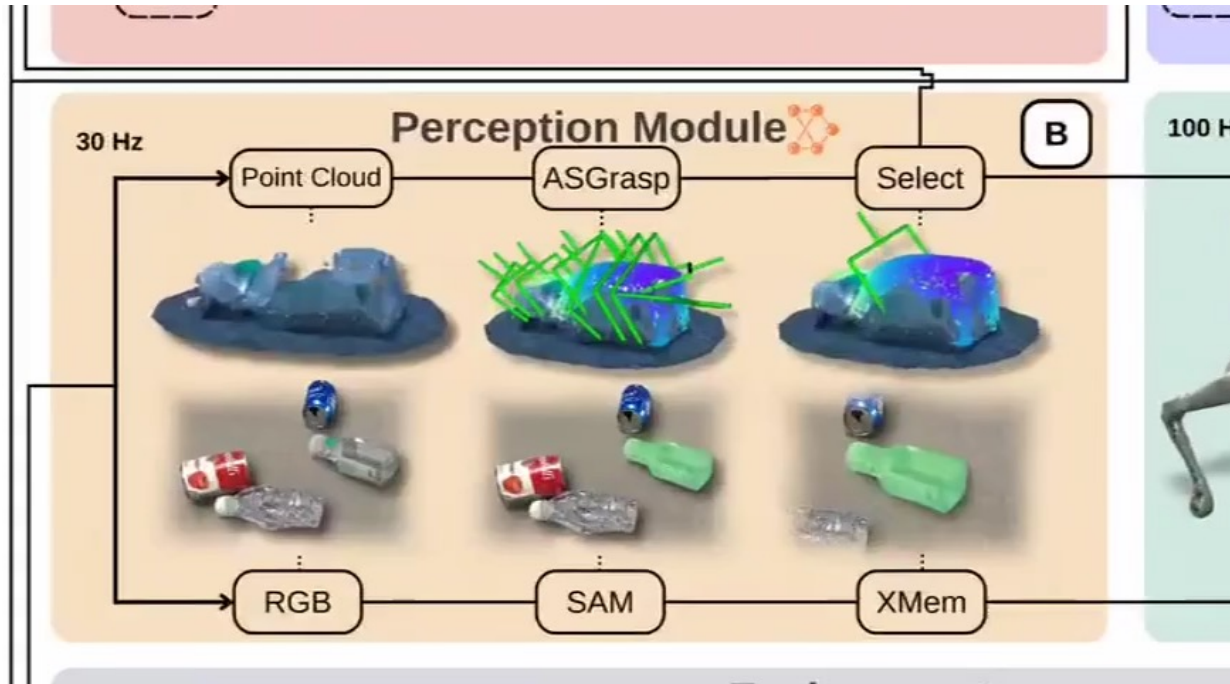
5D Command Locomotion

- linear velocity (x, y)
- yaw velocity
- height position (z)
- pitch position

Whole-body awareness

- randomize ee pose while training

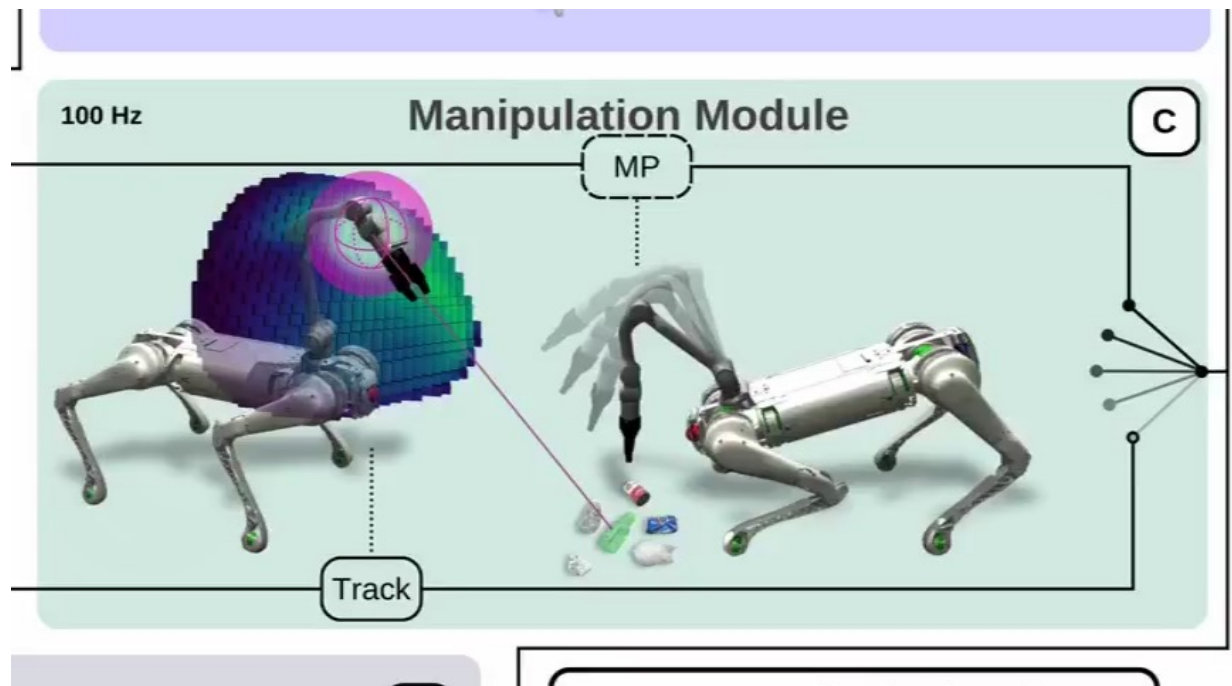
QuadWBG - Perception



Off-the-shelf Module

- Xmen for tracking
- AsGrasp for grasp detection

QuadWBG - Manipulation



Differential IK and online planning

- Continuous and smooth motion
- Adaptive to unexpected movements

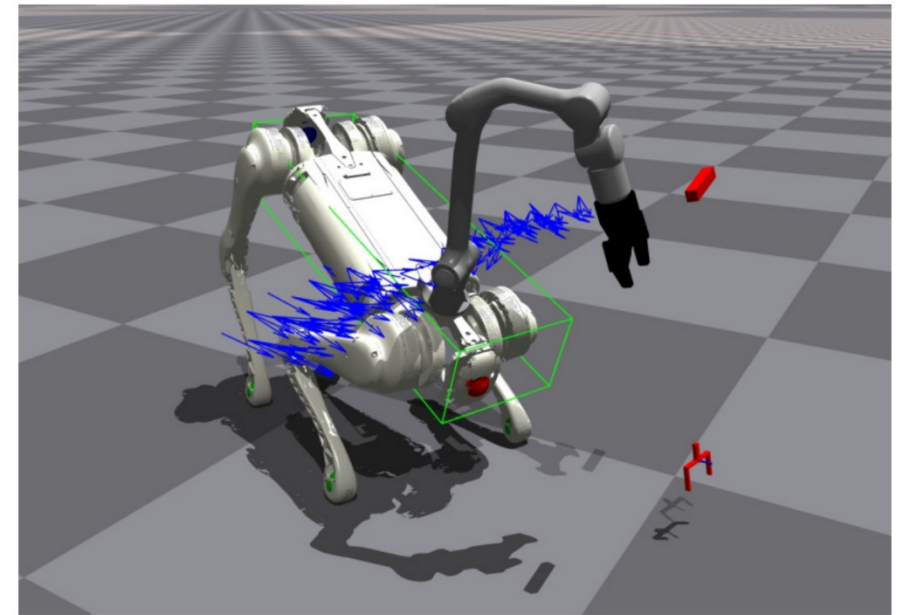
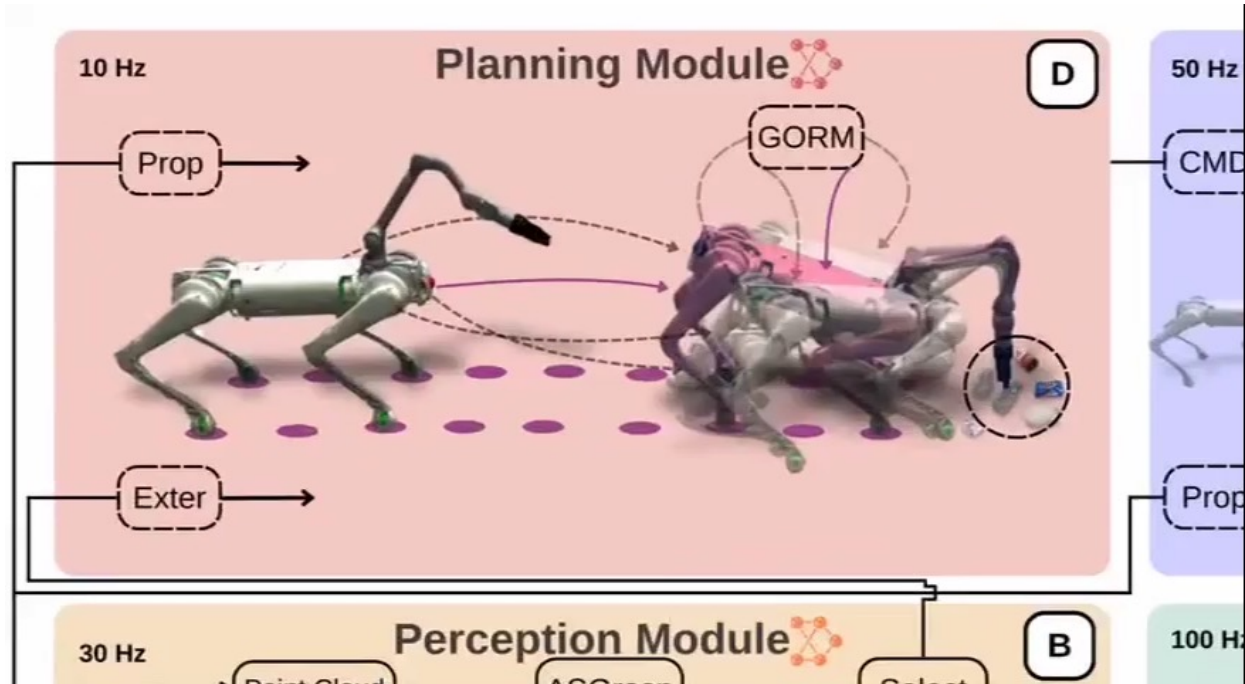
Parameterized tracking sphere

- Minimize motion blur on image
- Reduce loss of tracking

Reachability-based mode switching

- Statistics-based, accurate and stable

QuadWBG – High-level Planner



QuadWBG – High-level Planner

State

Observation	Dimension
goal pose	12
gravity vector	3
joint position	18
last action	5

Reward

$$r_{\text{GORM}} = \exp(-(\min(\mathbf{d}_{\text{GORM}}))^2)$$

$\mathbf{d}_{\text{GORM}}$: filtered base 6D pose distribution in TCP frame

Action

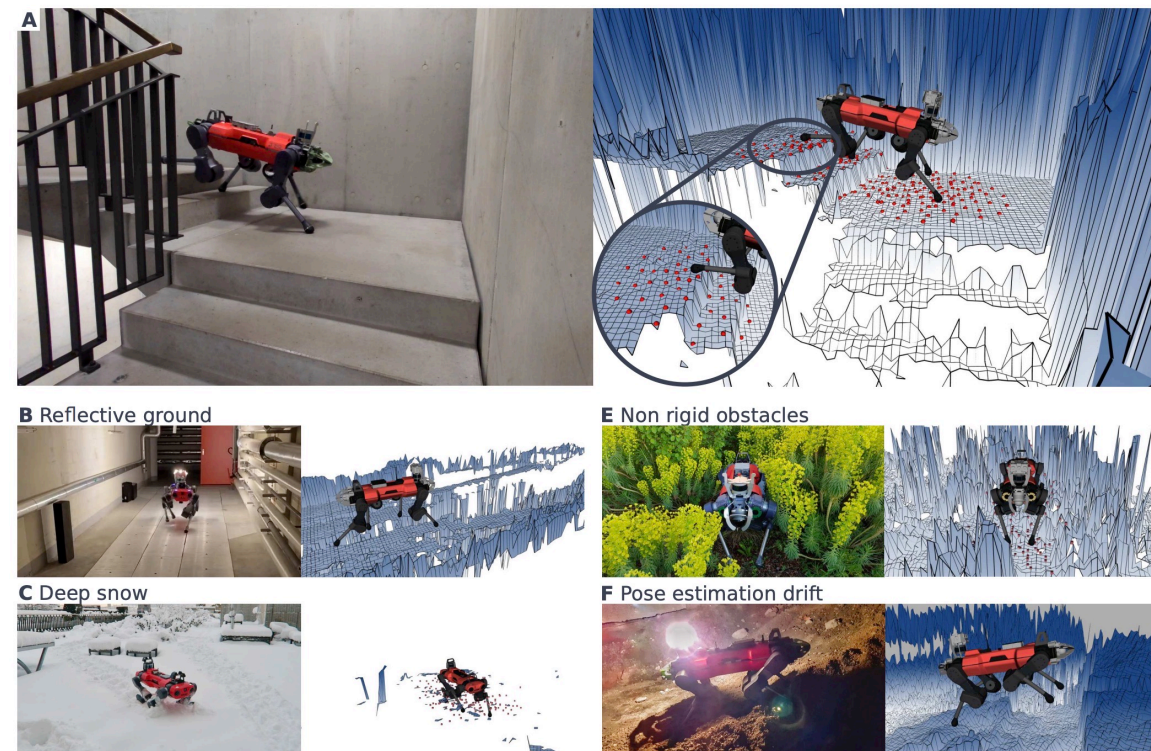
Action	Dimension
base height	1
base pitch	1
base linear velocity (x, y)	2
base yaw velocity	1

Other Exemplar Works



Robust to terrain
(only proprioception)

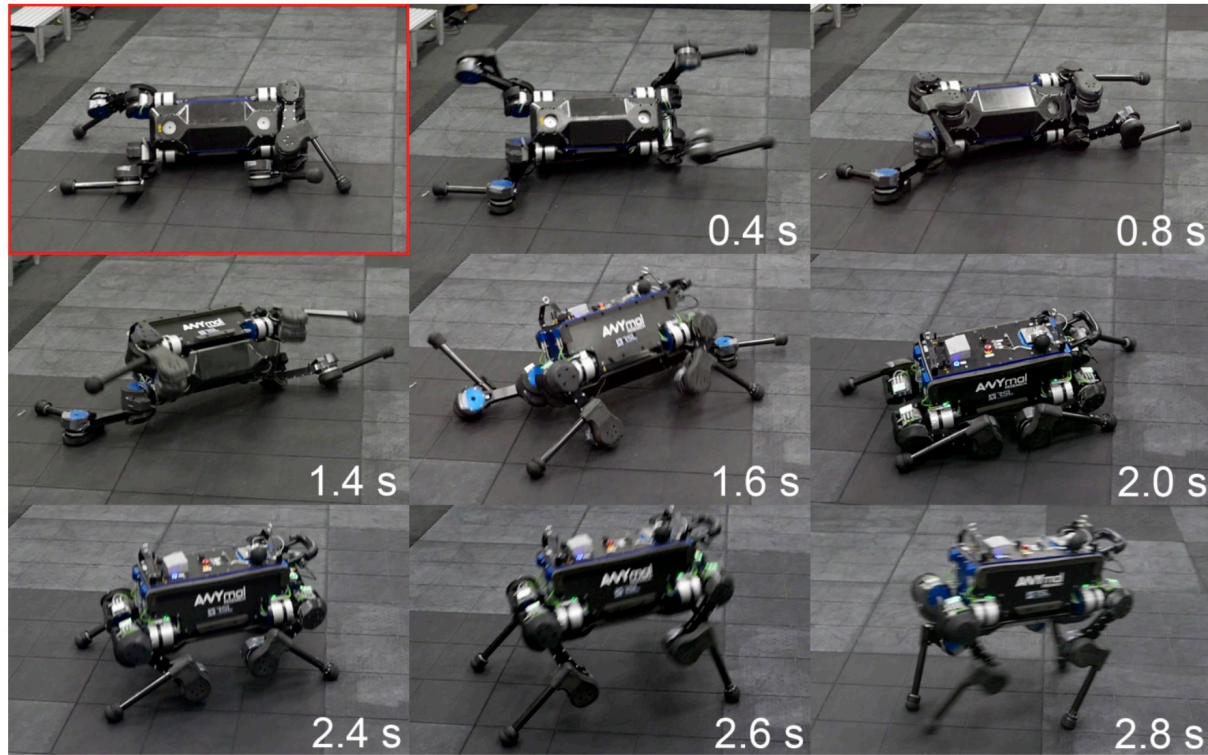
[Lee, Joonho, et al. 2020]



Robust to terrain
(perceive env by vision, radar)

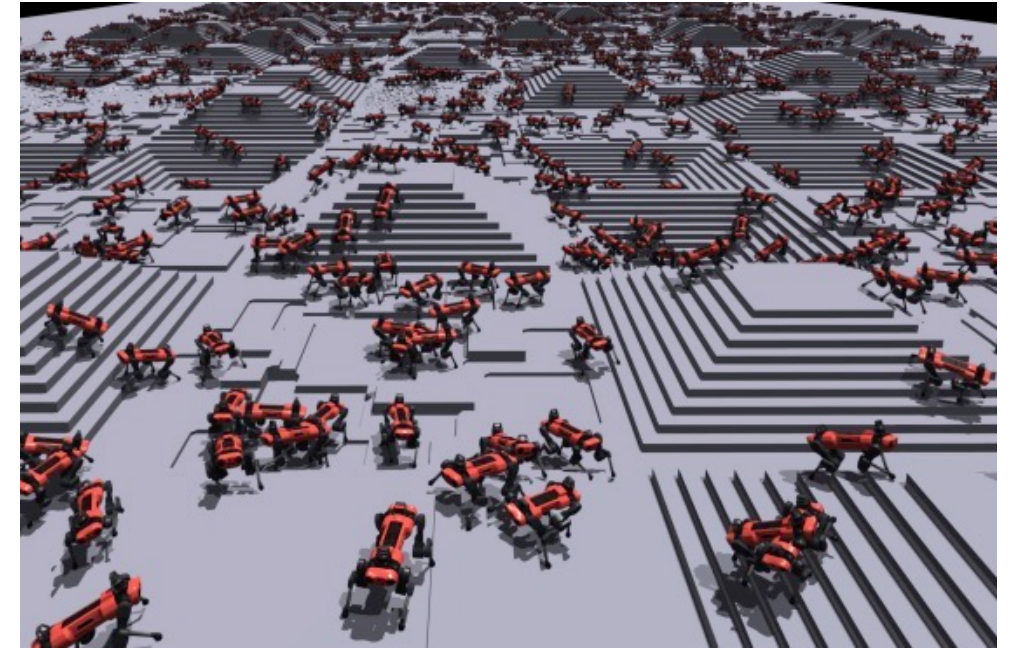
[TAKAHIRO MIKI et al. 2022]

Other Exemplar Works



Robust to a fall

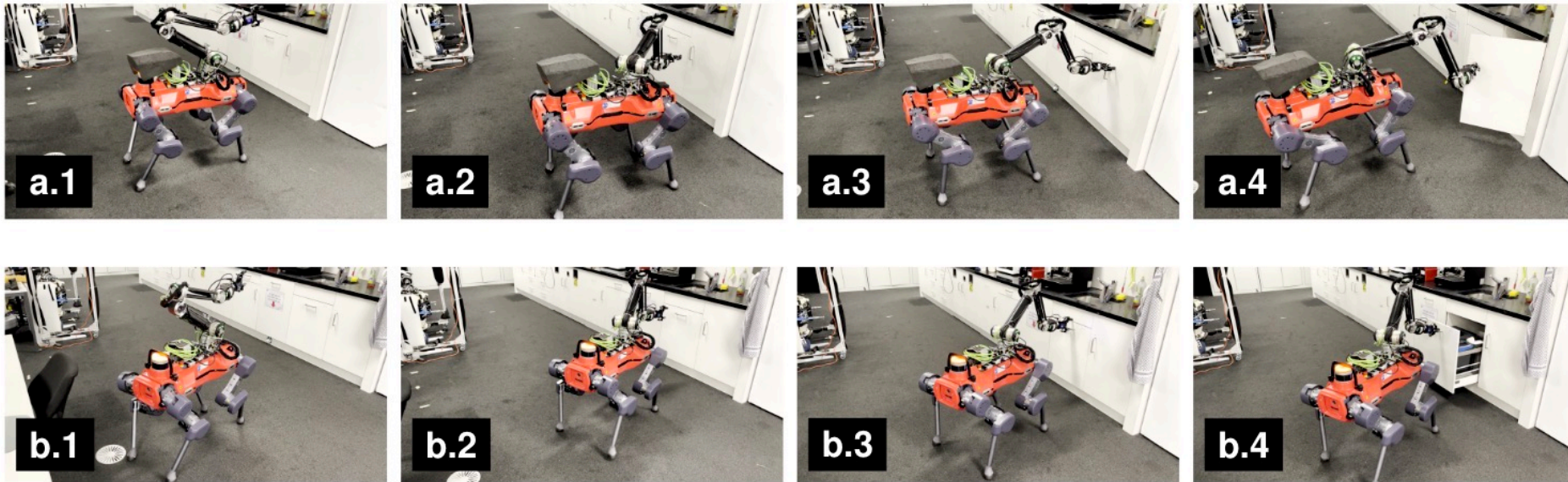
[Lee, Joonho, et al. 2019]



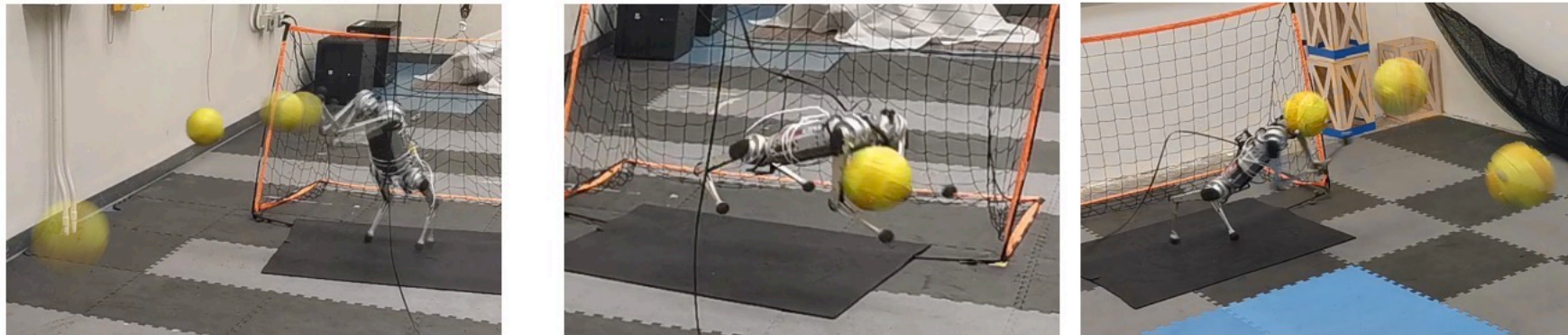
Learn fast

[Nikita Rudin et al. 2020]

Other Exemplar Works

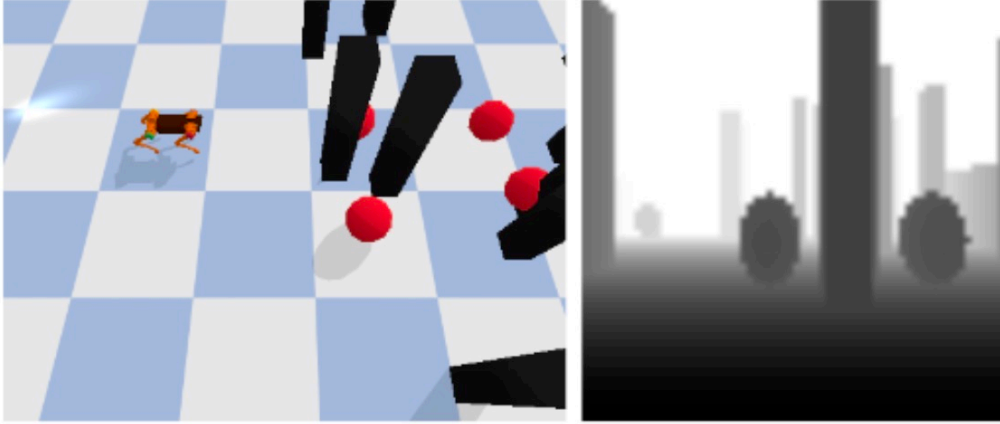


Downstream topic: mobile manipulation [Mayank Mittal et al. 2022]



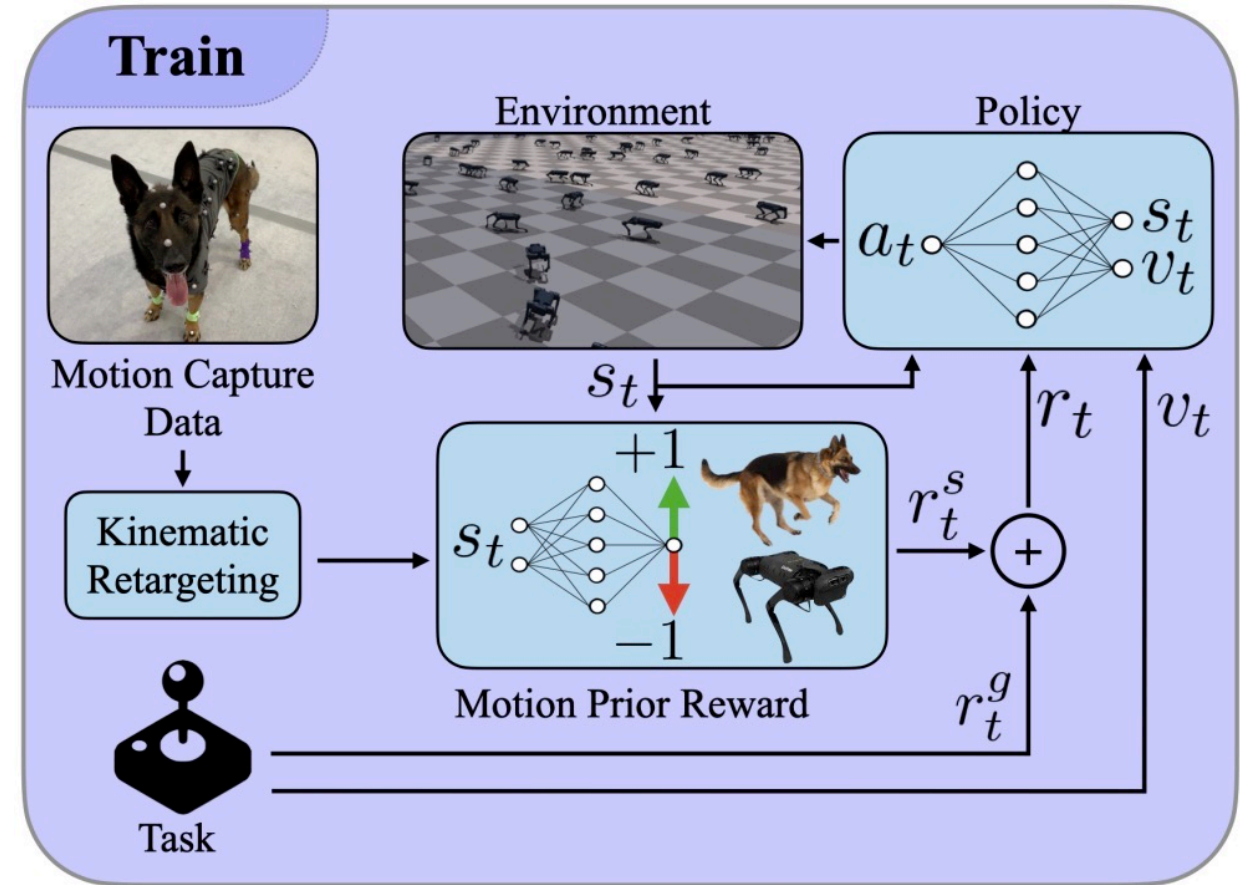
Downstream application: goalkeeper [Huang, Xiaoyu, et al. 2022]

Other Exemplar Works



Avoid obstacles

[Ruihan Yang et al. 2022]



Move naturally

[Alejandro Escontrela et al. 2022]

Challenge and Recent Works

Learning Vision-Guided Quadrupedal Locomotion End-to-End with Cross-Modal Transformers

Ruihan Yang^{*1}, Minghao Zhang^{*2}, Nicklas Hansen¹, Huazhe Xu³, Xiaolong Wang¹,

¹UC San Diego, ²Tsinghua University, ³UC Berkeley

ICLR 2022



Paper



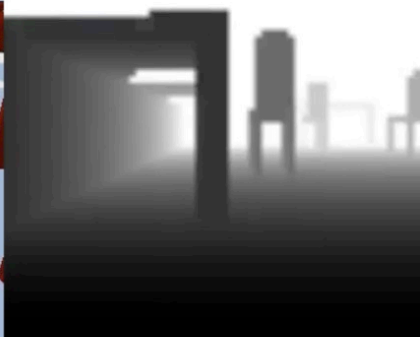
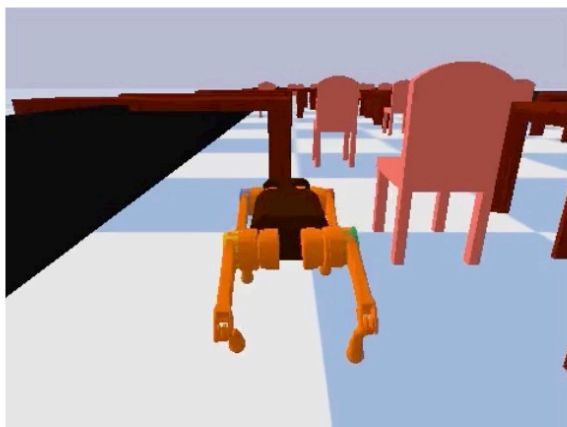
arXiv



Video



Code



Method

- End-to-end RL (PPO)
- Input
 - proprioceptive states
 - Temporal depth images
- Output
 - Action distribution or value

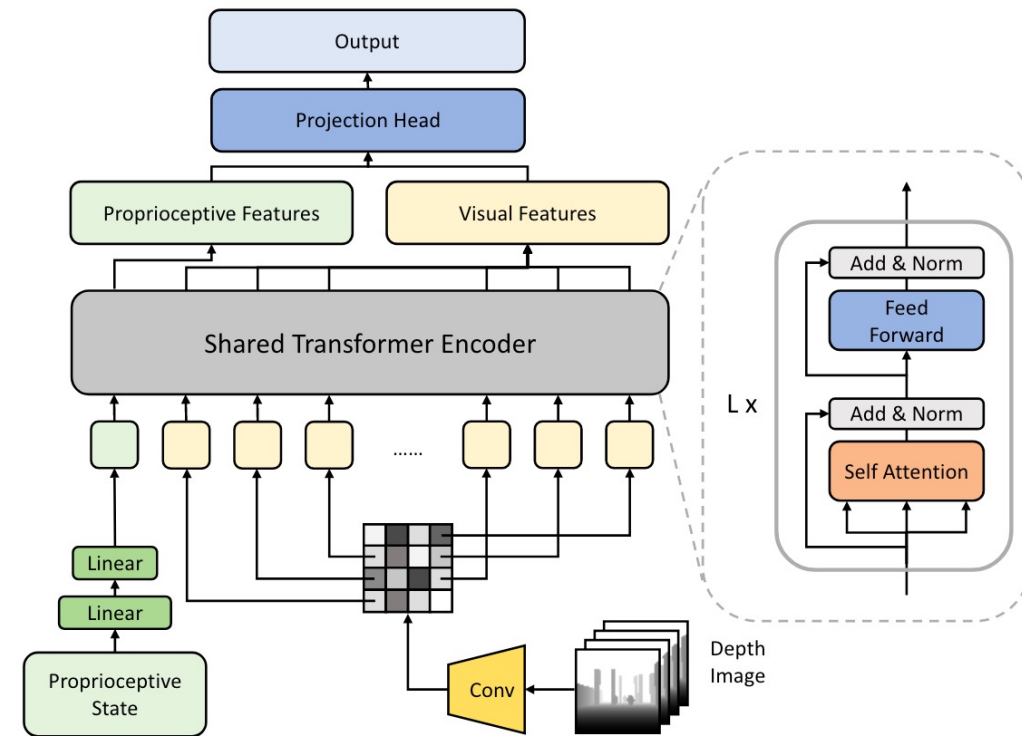
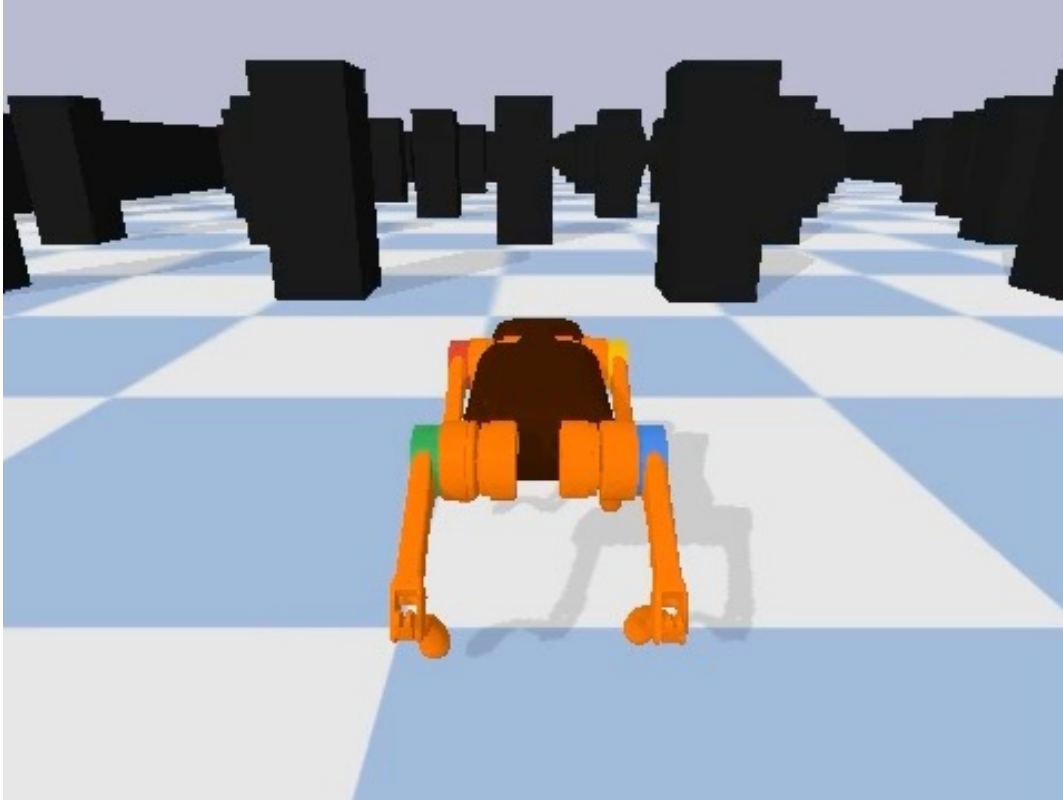
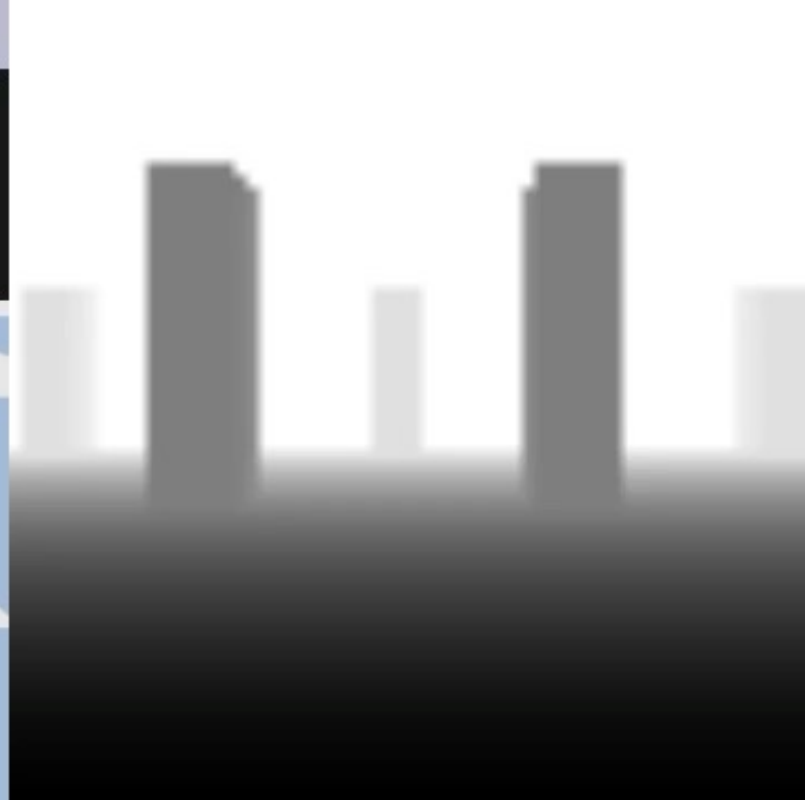


Figure 2: **Network Architecture.** We process proprioceptive states with a MLP and depth images with a ConvNet. We take proprioceptive embedding as a single token, split the spatial visual feature representation into $N \times N$ tokens and feed all tokens into the Transformer encoder. The output tokens are further processed by the projection head to predict value or action distribution.

Results in Simulation



Visualization



Depth observation

Results in Real World



Indoor



Outdoor

From Quadrupedal to Bipedal

- Key differences

Aspect	Quadrupedal	Bipedal
Structure	Low center-of-mass Small DOF	High center-of-mass Large DOF Complex contact modelling of feet
Learning	Traditional MPC / RL methods	Imitate mocap data (reference) Focus more on balancing
Task	Rough terrain traversal Load carrying	Human-like tasks: Loco-manipulation Dance

Bipedal – Use Reference Motion

Methods	Characteristic	Advantages and Disadvantages	Implementation Flow Chart
Residual learning	Add residential term to the known motor positions at the current time step.	A: <u>Fast convergence speed</u> D: <u>Require high-quality predefined reference</u>, limit to specific motions, and <u>lack robustness</u> to complicated terrains.	<pre> graph LR UC[User Commands] --> P[Policy π] Ref[Reference] --> P P --> R[Robots] R --> Sum((⊕)) Sum --> Ref </pre>
Guided learning	Mimic the predefined reference and directly specifies joint-level command.	A: Accelerate the learning process and <u>robust to the terrains</u>. D: <u>Limited to the predefined motions</u> and <u>lack adaptability</u> to unforeseen changes in environment.	<pre> graph LR UC[User Commands] --> P[Policy π] Ref[Reference] -.-> P P --> R[Robots] </pre>
Reference-free learning	Learn locomotion skills from scratch without any prior knowledge.	A: High potential for gait exploration, <u>high robust</u> to the complicated terrain. D: <u>Requires intensive reward shaping</u> for gait patterns and relatively expensive computational training resources.	<pre> graph LR UC[User Commands] --> P[Policy π] P --> R[Robots] </pre>

Bipedal – Use Reference Motion

ExBody²: Advanced Expressive Humanoid Whole-Body Control

Mazeyu Ji^{*1}

Xuanbin Peng^{*1}

Fangchen Liu²

Jialong Li¹

Ge Yang³

Xuxin Cheng^{1,†}

Xiaolong Wang^{1,4,†}

¹UC San Diego

²UC Berkeley

³MIT

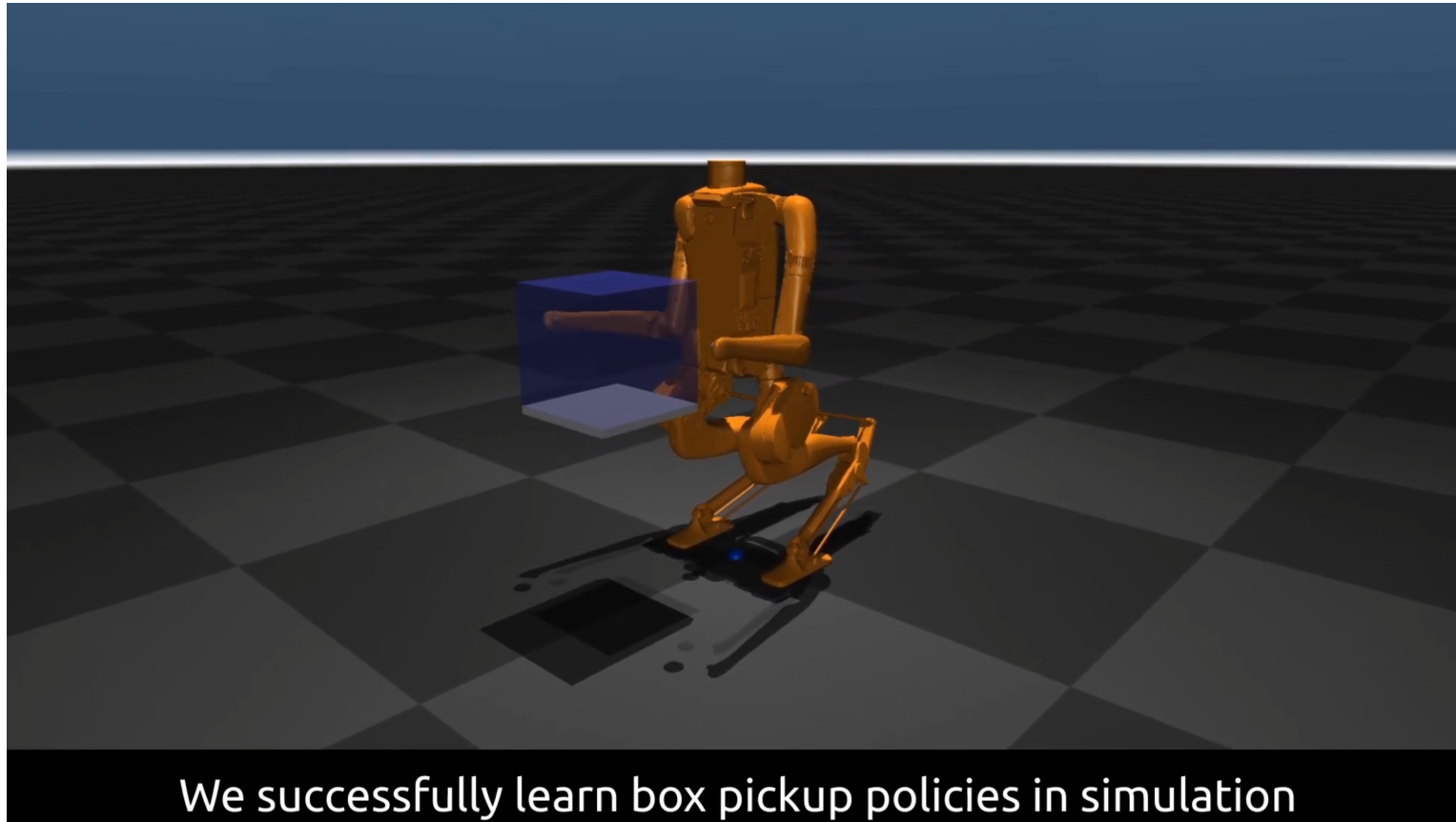
⁴NVIDIA



Bipedal – Loco-Manipulation

Sim-to-Real Learning for Humanoid Box Loco-Manipulation

Jeremy Dao, Helei Duan, Alan Fern



We successfully learn box pickup policies in simulation

Bipedal – Contact Handling

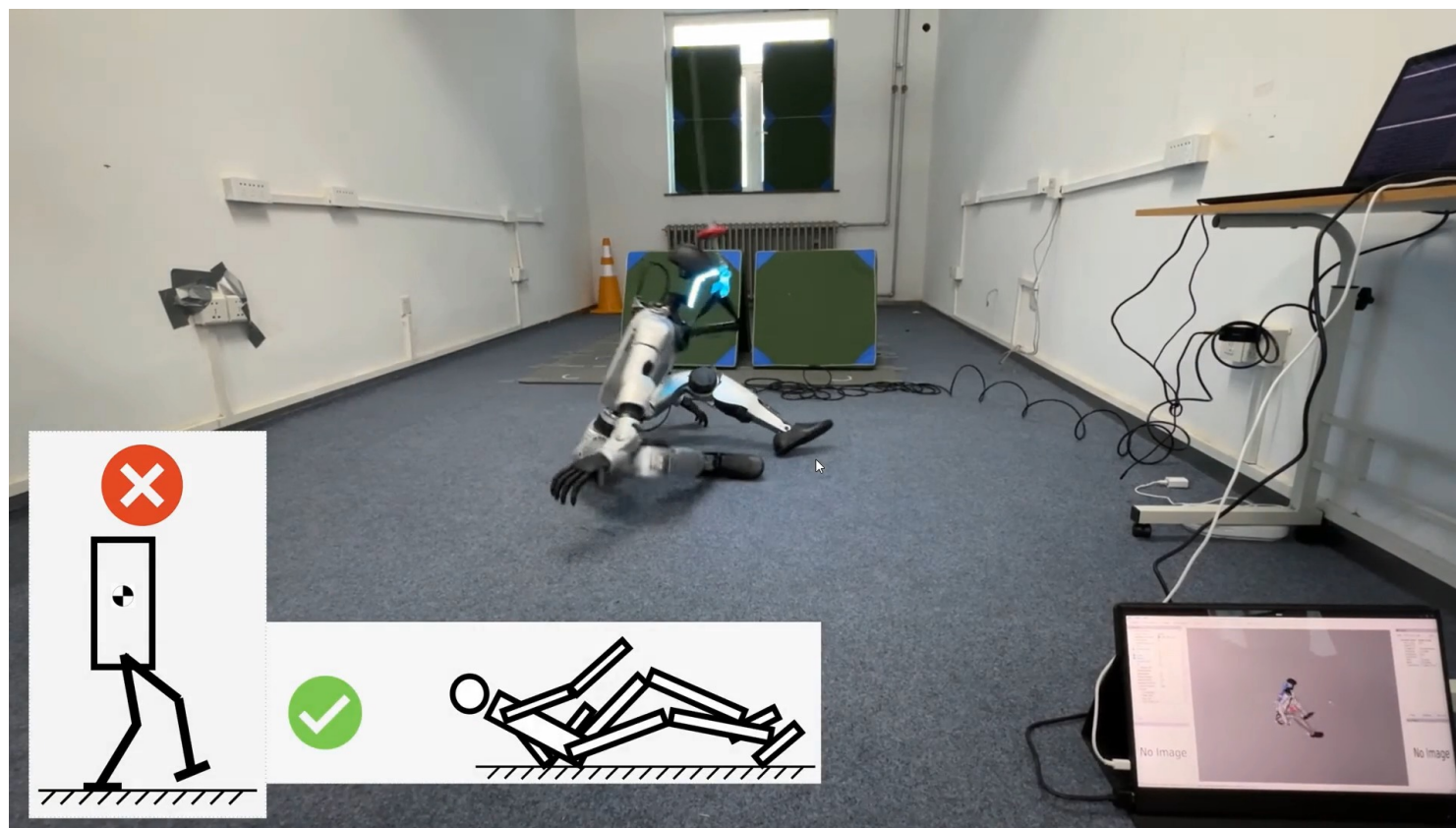
Embrace Collisions: Humanoid Shadowing for Deployable Contact-Agnostic Motions

Ziwen Zhuang

IIS, Tsinghua University
Shanghai Qi Zhi Institute

Hang Zhao

IIS, Tsinghua University
Shanghai Qi Zhi Institute



Sim-to-real

- Ways to mitigate sim-to-real gap
 - Reward design
 - Action smoothness, action rate
 - Domain Randomization
 - Dynamics: center of mass, mass, friction
 - Other: latency, image augmentation
 - Domain Adaptation
 - System Identification
 - Model actuator dynamics with real data
 - Better contact model in simulation

Difficulty

- Open-source is a problem
 - Many companies don't open algorithm
 - Hard to reimplement even if open-sourced algorithm
- Both MPC and RL needs careful tuning
 - Each gait needs to tune a model (especially for MPC)
- Robustness issue and sim-to-real gap

Future Directions

- Combining control and learning
 - Using trajectory planner to guide the low-level RL-based controller
 - Using RL to calculate reference key points (e.g. foothold location) for MPC
 - ...
- Better Training Strategy
 - Make training robust to the coefficients of reward terms
- Safety
 - Safe RL
- Mobile-manipulation
 - Make upper-body better coordinate with lower-body

Additional Resources

- Codebase
 - Unitree RL GYM: https://github.com/unitreerobotics/unitree_rl_gym
 - Legged GYM: https://github.com/leggedrobotics/legged_gym
 - MuJoCo Playground: https://github.com/google-deepmind/mujoco_playground
- Papers
 - Quadrupedal Locomotion
 - Learning-based legged locomotion: state of the art and future perspectives
 - Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning
 - DreamWaQ: Learning Robust Quadrupedal Locomotion With Implicit Terrain Imagination via Deep Reinforcement Learning
 - Legged Locomotion in Challenging Terrains using Egocentric Vision
 - ANYmal Parkour: Learning Agile Navigation for Quadrupedal Robots
 - Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior

Additional Resources

- Papers

- Quadrupedal Locomotion

- Learning quadrupedal locomotion on deformable terrain
 - Data Efficient Reinforcement Learning for Legged Robots

- Quadrupedal Loco-Manipulation

- Visual Whole-Body Control for Legged Loco-Manipulation

- Bipedal Locomotion

- Deep Reinforcement Learning for Bipedal Locomotion: A Brief Survey
 - ExBody2: Advanced Expressive Humanoid Whole-Body Control
 - Reinforcement Learning for Versatile, Dynamic, and Robust Bipedal Locomotion Control
 - Toward Understanding Key Estimation in Learning Robust Humanoid Locomotion
 - Berkeley Humanoid: A Research Platform for Learning-based Control



Introduction to Embodied AI

Next week:
Lecture 11, Locomotion

