RMA: Rapid Motor Adaptation for Legged Robots

Ashish Kumar, Zipeng Fu, Deepak Pathak, Jitendra Malik

Presented by

Youren Zhang | yourenz@umich.edu

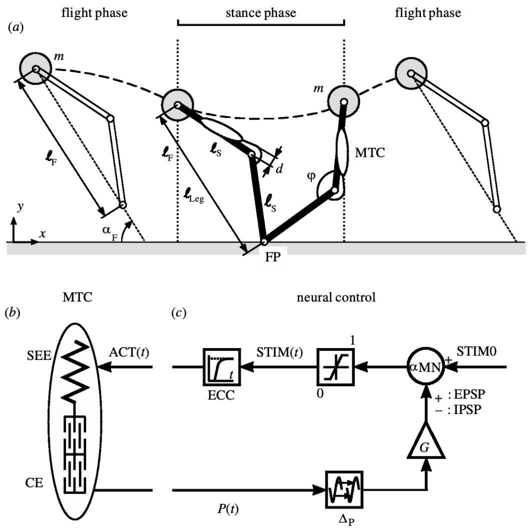
LEGGED ROBOTS

Legged robots are a type of mobile robot which use ***articulated limbs***, such as leg mechanisms, to provide locomotion.

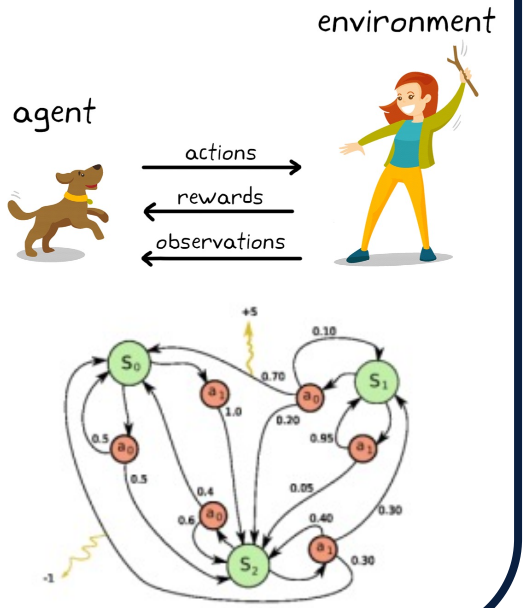
Quadruped robots

CONTROLLER

Traditional Methods

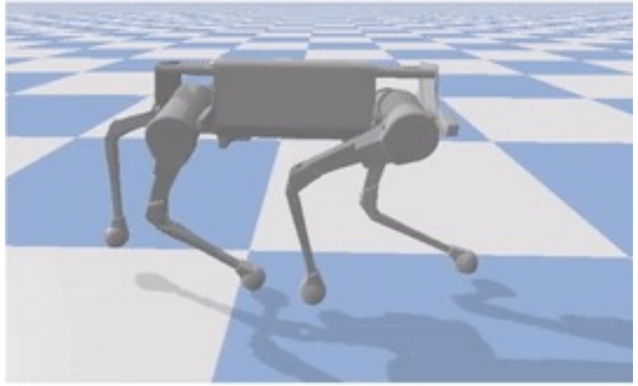


Learning-based Methods



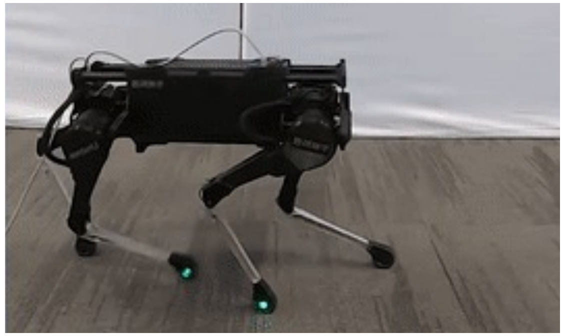
- Require considerable expertise on the part of the human designer

- Train in **simulation**, then transfer to the real-world using sim-to-real techniques



Generalization

Simulation → Reality



- The physical robot and its model in the simulator differ significantly
- Real-world terrains vary considerably
- The simulator fails to accurately capture the physics of the real world

For quadruped robots

To solve generalization problem, the authors proposed

Rapid Motor Adaptation

Learned entirely in ***simulation*** (why?) without using any domain knowledge

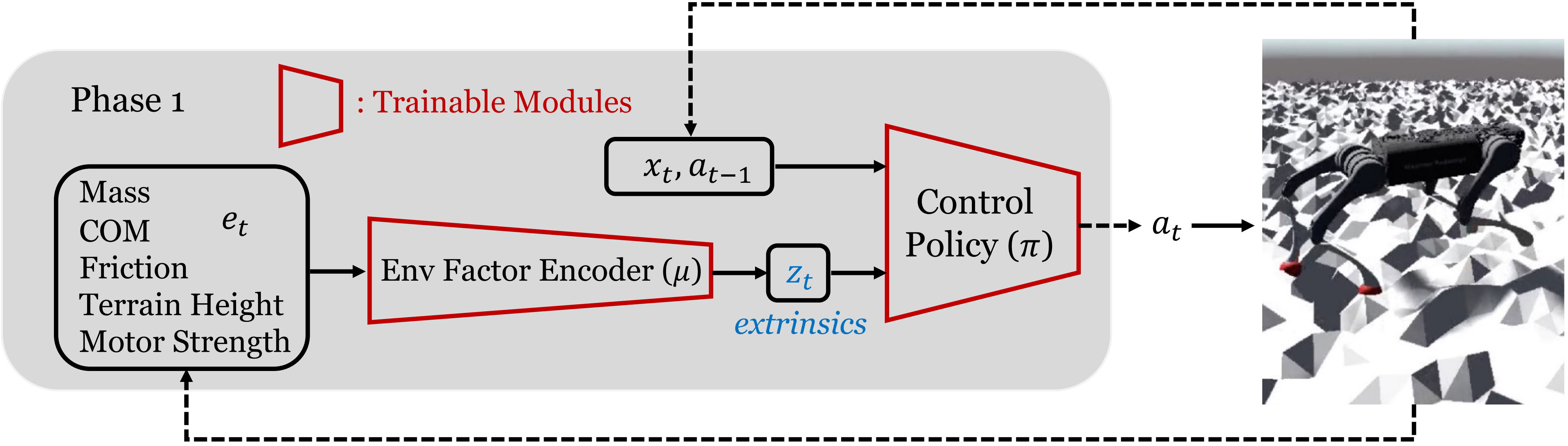
Deploy ***without*** fine-tuning

Rapid Motor Adaptation

Train in Simulation

Phase 1: Jointly train policy π and environmental factor encoder μ via Reinforcement Learning in simulation

$$a_t = \pi(x_t, a_{t-1}, z_t) = \pi(x_t, a_{t-1}, \mu(e_t))$$



Train in Simulation

Phase 1: Jointly train policy π and environmental factor encoder μ via Reinforcement Learning in simulation

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

where $\tau = \{(x_0, a_o, r_0), (x_1, a_1, r_1), \dots\}$ is the trajectory of the agent when executing policy π, γ is the hyperparameter, which is set to 0.998 according to the supplementary.

(More in discussion)
Follow a designed *curriculum*
The reward function r_t is the *weighted* sum of

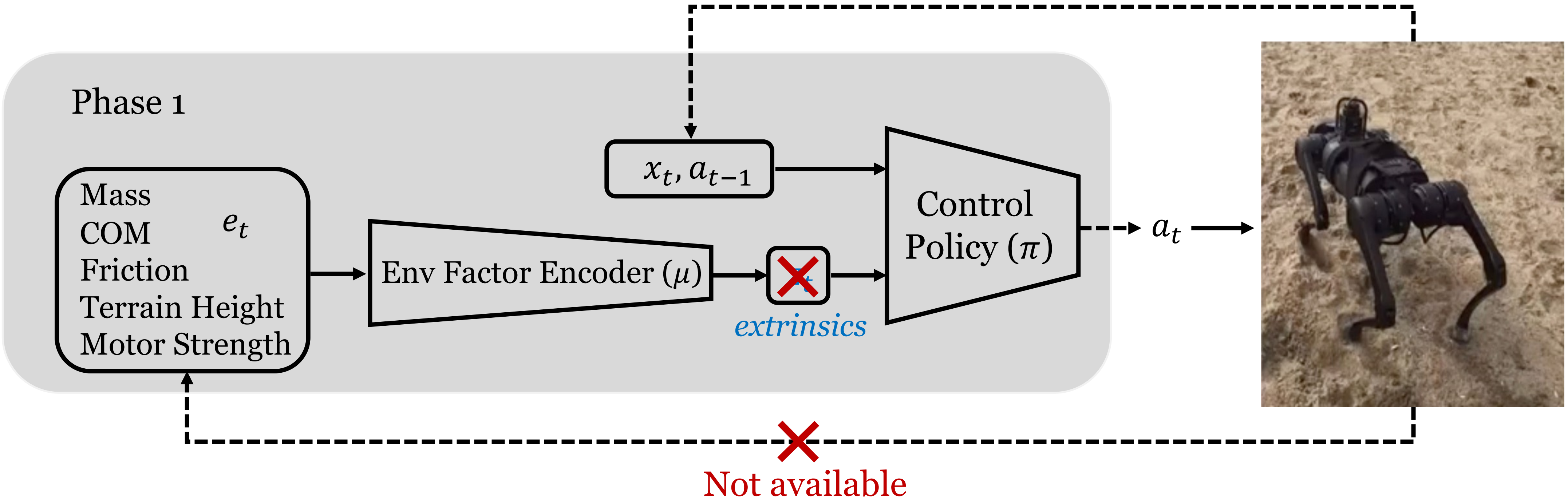
1) Forward: $\min(v_x^t, 0.35)$
2) Lateral Movement and Rotation: $-\|v_y^t\|^2 - \|\omega_{\mathtt{yaw}}^t\|^2$
3) Work: $-|\boldsymbol{\tau}^T \cdot (\mathbf{q}^t - \mathbf{q}^{t-1})|$
4) Ground Impact: $-\|\mathbf{f}^t - \mathbf{f}^{t-1}\|^2$
5) Smoothness: $-\|\boldsymbol{\tau}^t - \boldsymbol{\tau}^{t-1}\|^2$
6) Action Magnitude: $-\|\mathbf{a}^t\|^2$
7) Joint Speed: $-\|\dot{\mathbf{q}}^t\|^2$
8) Orientation: $-\|\boldsymbol{\theta}_{\mathtt{roll,\ pitch}}^t\|^2$
9) Z Acceleration: $-\|v_z^t\|^2$
10) Foot Slip: $-\|\mathrm{diag}(\mathbf{g}^t) \cdot \mathbf{v_f}^t\|^2$

Penalize for jerky and inefficient motions

Bioenergetics-inspired, learn walking policies ***without*** using any reference demonstrations by minimizing work and ground impact.

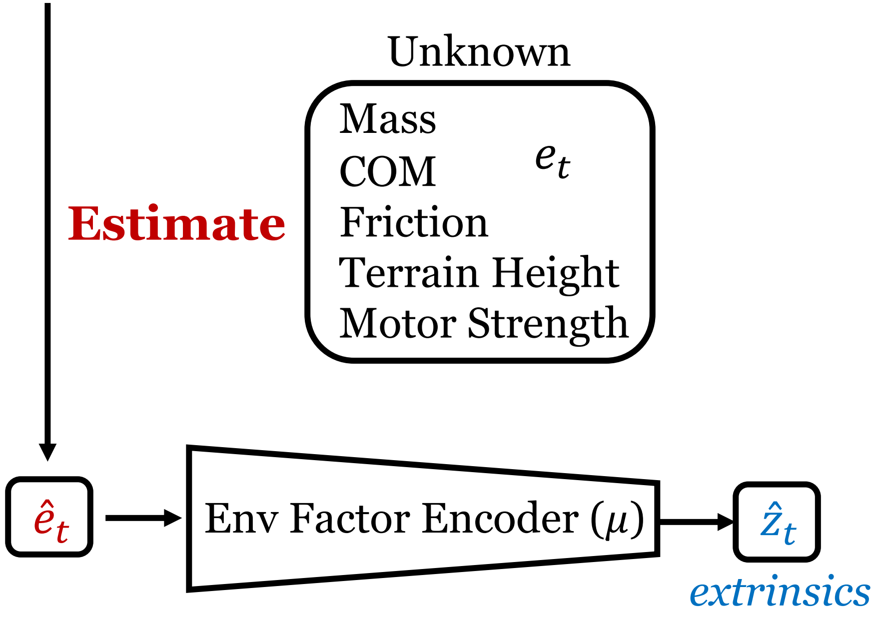
How to Deploy

However, environmental factors are ***not*** available when deploying.



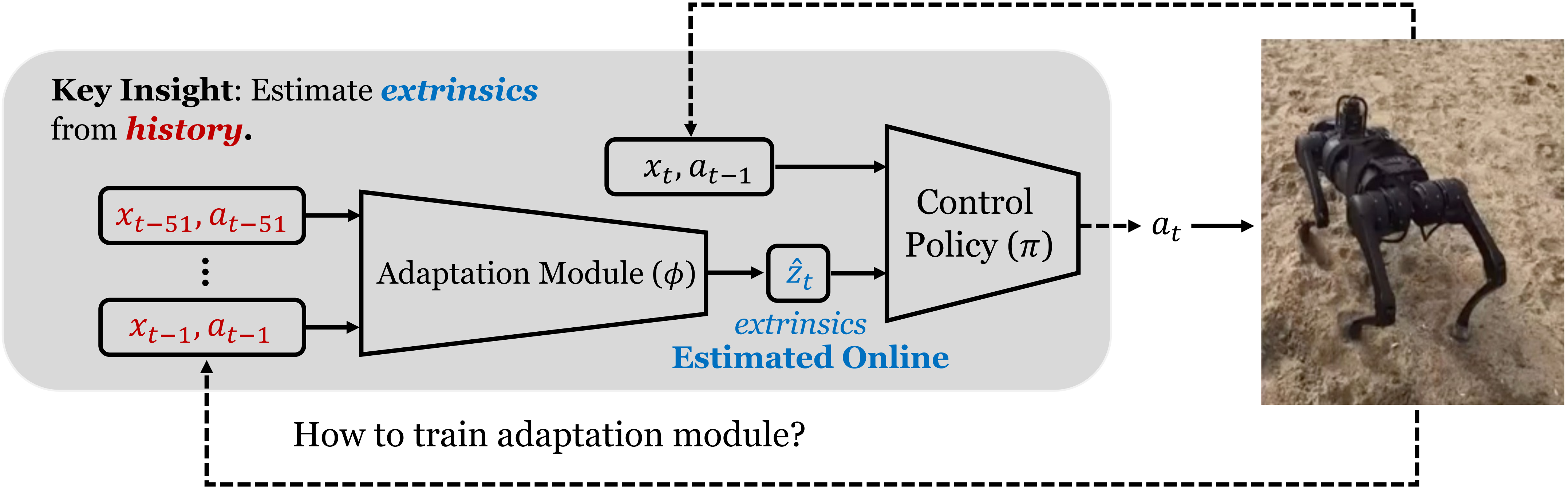
How to Deploy

The task of **System Identification** **is Very Hard!!**



Adaptation Module

Instead of system identification, **directly estimate the *extrinsics*.**

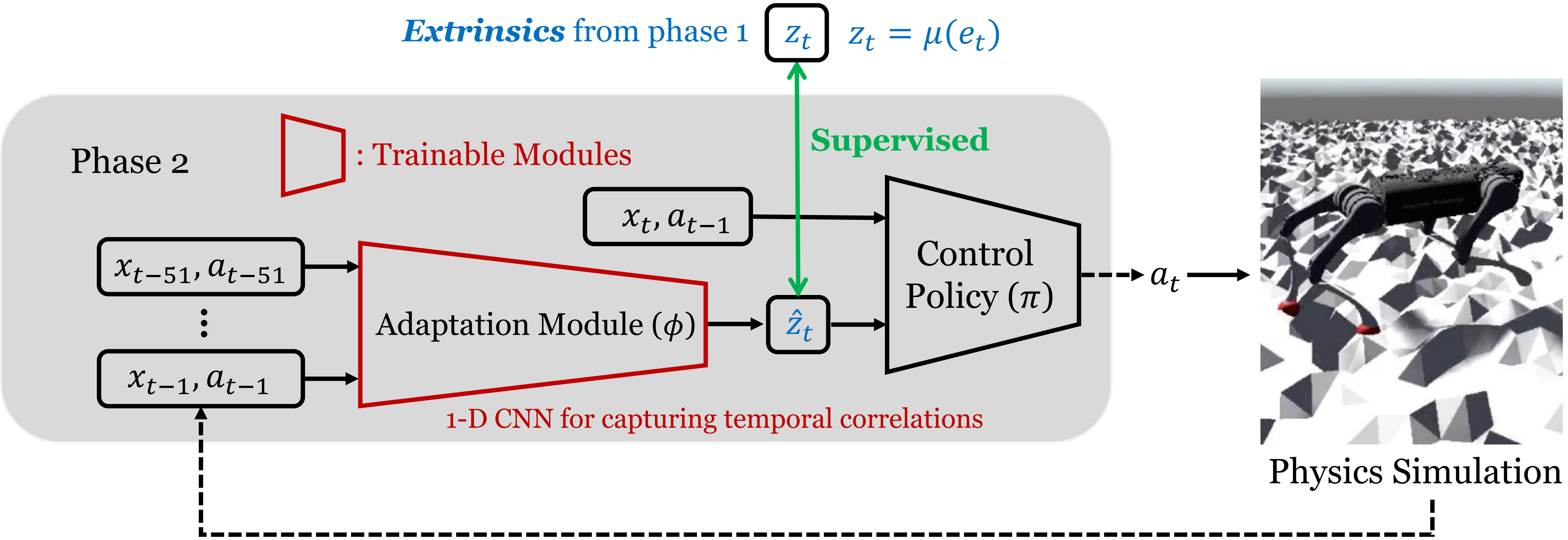


How to train adaptation module?

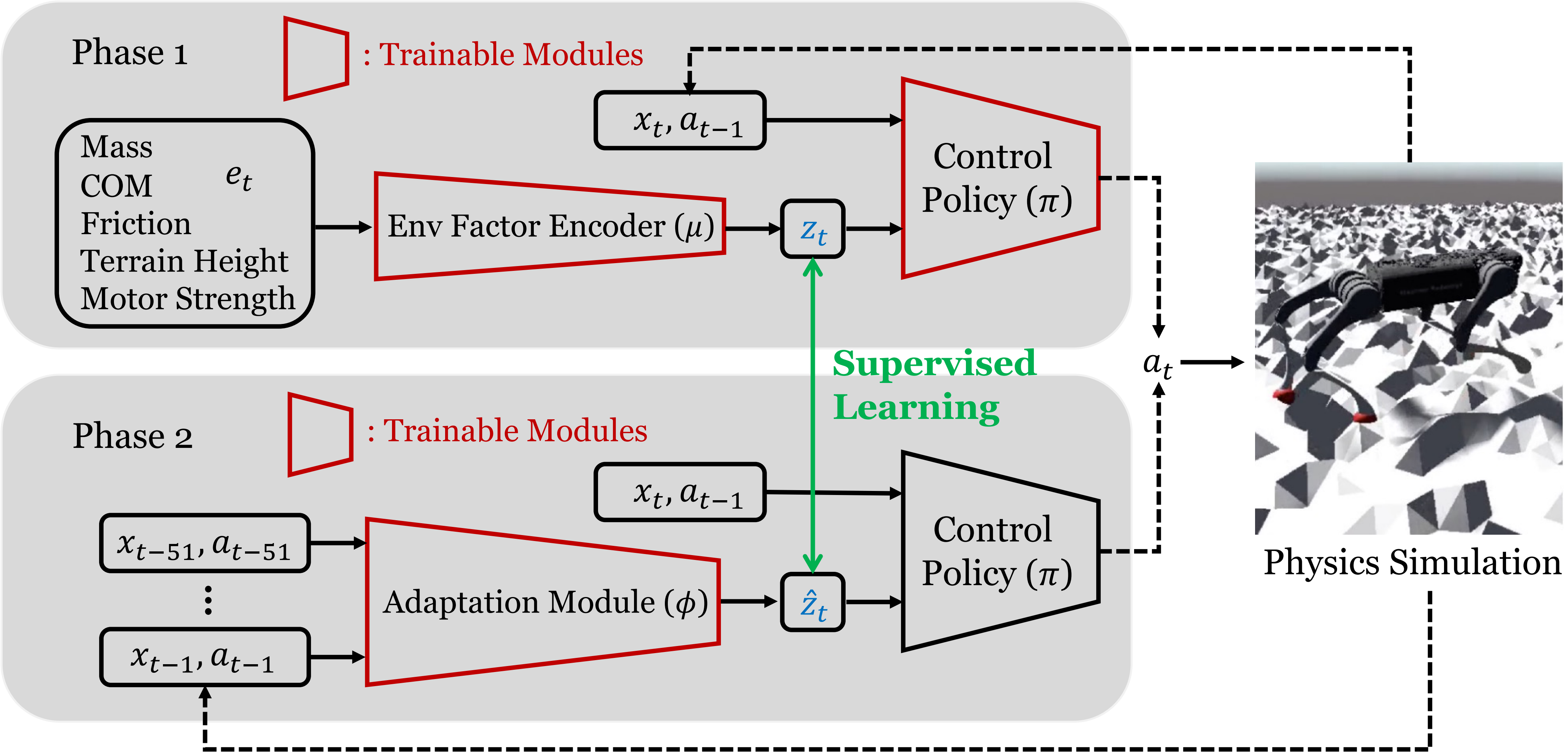
Training Adaptation Module

Phase 2: Train adaptation module ϕ via Supervised Learning in simulation, $k = 50$ (0.5s) in experiments

$$\hat{z}_t = \phi(x_{t-k:t-1}, a_{t-k:t-1})$$



Training Scheme



Training Algorithm

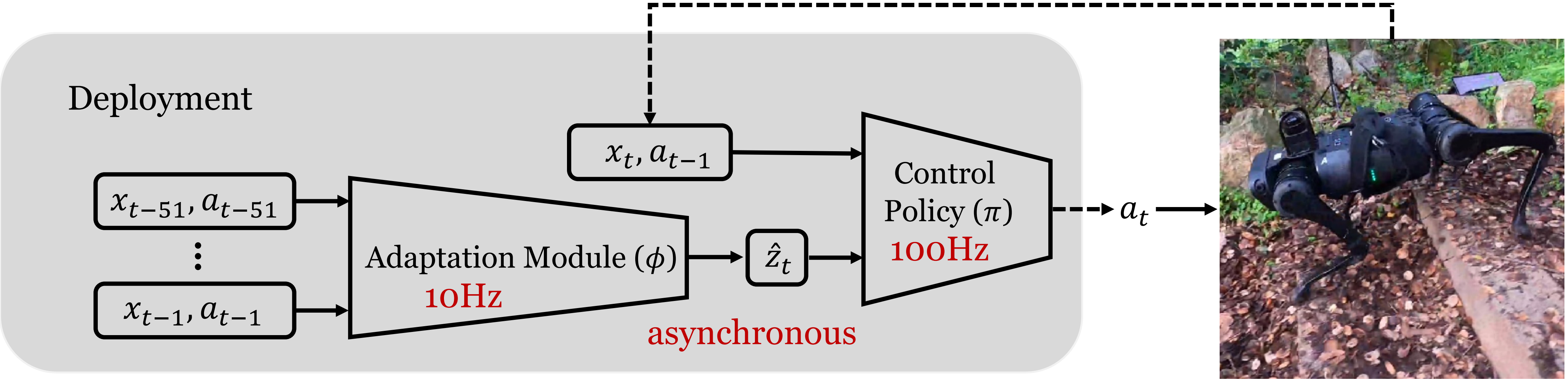
Phase 1 Randomly initialize the base policy π;
Randomly initialize the environmental factor encoder
μ; Empty replay buffer D_1;
for $0 \leq \text{itr} \leq N^1_{\text{itr}}$ **do**
  for $0 \leq i \leq N_{\text{env}}$ **do**
    $x_0, e_0 \leftarrow$ envs[i].reset();
    for $0 \leq t \leq T$ **do**
      $z_t \leftarrow \mu(e_t)$;
      $a_t \leftarrow \pi(x_t, a_{t-1}, z_t)$;
      $x_{t+1}, e_{t+1}, r_t \leftarrow$ envs[i].step(a_t);
      Store $((x_t, e_t), a_t, r_t, (x_{t+1}, e_{t+1}))$ in D_1;
    end
  end
  Update π and μ using PPO [48];
  Empty D_1;
end

Phase 2 Randomly initialize the adaptation module ϕ
parameterized by θ_ϕ; Empty mini-batch D_2;
for $0 \leq \text{itr} \leq N^2_{\text{itr}}$ **do**
  for $0 \leq i \leq N_{\text{env}}$ **do**
    $x_0, e_0 \leftarrow$ envs[i].reset();
    for $0 \leq t \leq T$ **do**
      $\hat{\mathbf{z}}_\mathbf{t} \leftarrow \phi(x_{t-k:k}, a_{t-k-1:k-1})$;
      $z_t \leftarrow \mu(e_t)$;
      $a_t \leftarrow \pi(x_t, a_{t-1}, \hat{\mathbf{z}}_\mathbf{t})$;
      $x_{t+1}, e_{t+1}, \_ \leftarrow$ envs[i].step(a_t);
      Store $(\hat{z}_t, z_t)$ in D_2;
    end
  end
  $\theta_\phi \leftarrow \theta_\phi - \lambda_{\theta_\phi} \nabla_{\theta_\phi} \frac{1}{T N_{\text{env}}} \sum \|\hat{z}_t - z_t\|^2$;
  Empty D_2;
end

Deploy in Real-World

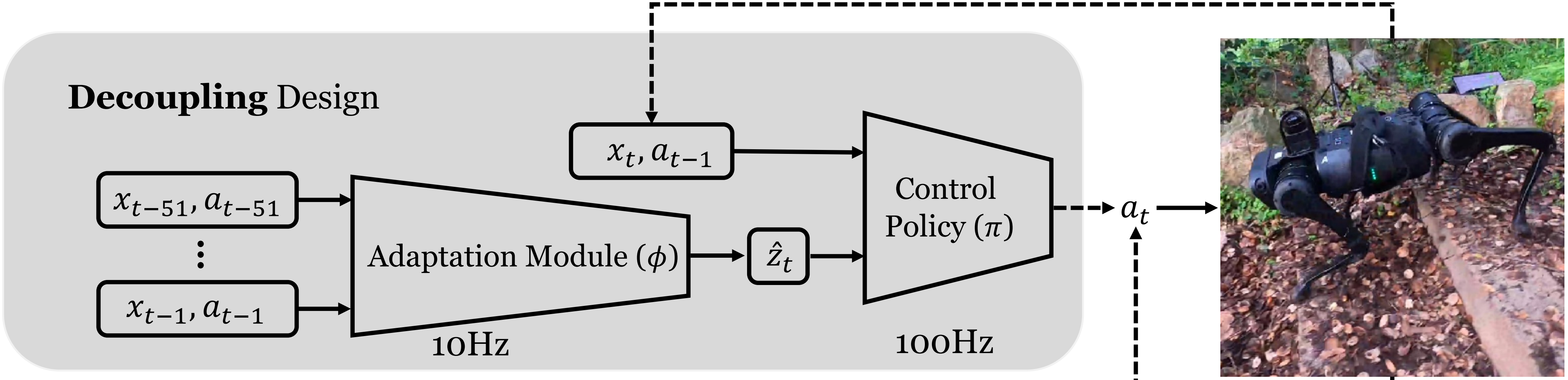
Deployment: The adaption module and the control policy run ***asynchronously***.

control policy uses most recent $\hat{z}_t$



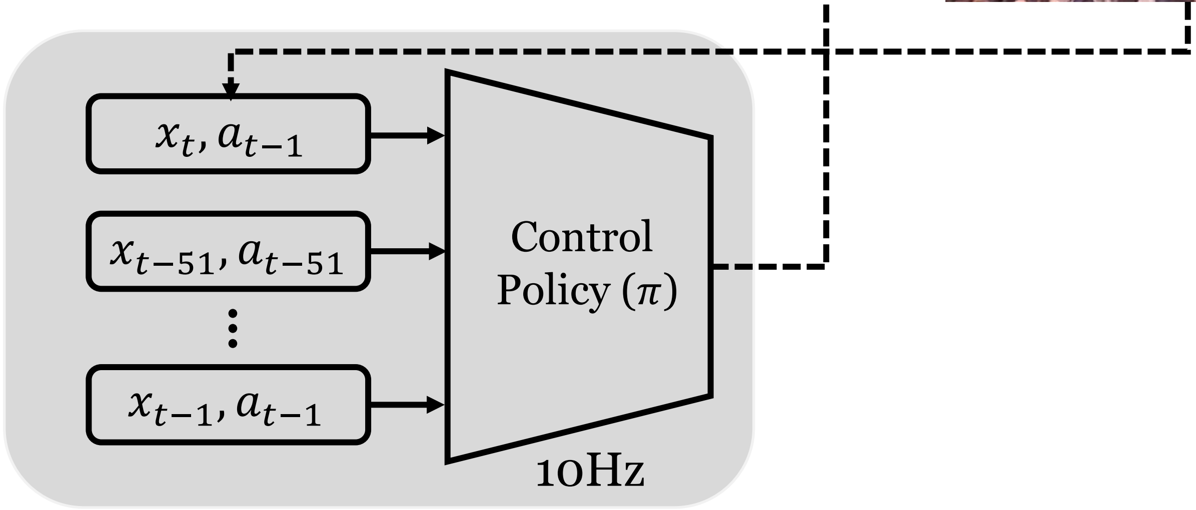
Intuition: $\hat{z}_t$ changes relatively infrequently in the real-world.

Necessity of Adaptation Module



Why decoupling?

- **Unnatural** gaits and **poor** performance in simulation.
- Can only run at **10Hz** on the on-board processor.
- Asynchronous design is critical for a **seamless** deployment, without needing calibration.



Environmental Variation

Parameters	Training Range	Testing Range
Friction	[0.05, 4.5]	[0.04, 6.0]
K_p	[50, 60]	[45, 65]
K_d	[0.4, 0.8]	[0.3, 0.9]
Payload (Kg)	[0, 6]	[0, 7]
Center of Mass (cm)	[-0.15, 0.15]	[-0.18, 0.18]
Motor Strength	[0.90, 1.10]	[0.88, 1.22]
Re-sample Probability	0.004	0.01

TABLE I: Ranges of the environmental parameters.

Experiment

Baselines

- **A1 Controller**: Default controller
- Robustness through Domain Randomization (**Robust**): The base policy is trained without z_t to be robust to the variations in the training range
- Expert Adaptation Policy (**Expert**): In simulation, we can use the true value of the extrinsics vector z_t. This is an upper bound to the performance of RMA.
- **RMA w/o Adaptation**: Run adaptation module for the first timestamp and then *freeze* it.
- **System Identification**: Directly predict the environmental factor e^t.
- Advantage Weighted Regression for Domain Adaptation (**AWR**): Optimize z^t offline using AWR by using real-world rollouts of the policy in the testing environment.

Rapid Motor Adaptation for Legged Robots

Ashish Kumar
UC Berkeley

Zipeng Fu
CMU

Deepak Pathak
CMU

Jitendra Malik
UC Berkeley/FAIR

Robotics: Science and Systems 2021

Rapid Motor Adaptation for Legged Robots

Ashish Kumar
UC Berkeley

Zipeng Fu
CMU

Deepak Pathak
CMU

Jitendra Malik
UC Berkeley/FAIR

Robotics: Science and Systems 2021

Gait pattern

Torque of knee

Components of extrinsics

Rapid Motor Adaptation for Legged Robots

Ashish Kumar
UC Berkeley

Zipeng Fu
CMU

Deepak Pathak
CMU

Jitendra Malik
UC Berkeley/FAIR

Robotics: Science and Systems 2021

EXPERIMENT

Results in simulation

	Success (%)	TTF	Reward	Distance (m)	Samples	Torque	Smoothness	Ground Impact
Robust [52, 40]	62.4	0.80	4.62	1.13	0	527.59	122.50	4.20
SysID [57]	56.5	0.74	4.82	1.17	0	565.85	149.75	4.03
AWR [41]	41.7	0.65	4.17	0.95	40k	599.71	162.60	4.02
RMA w/o Adapt	52.1	0.75	4.72	1.15	0	524.18	106.25	4.55
RMA	73.5	0.85	5.22	1.34	0	500.00	92.85	4.27
Expert	76.2	0.86	5.23	1.35	0	485.07	85.56	3.90

TABLE II: **Simulation Testing Results:** We compare the performance of our method to baseline methods in simulation. Our train and test settings are listed in Table I. We resample the environment parameters within an episode with a re-sampling probability of 0.01 per step during testing. Baselines and metrics are defined in Section V. The numbers reported are averaged over 3 randomly initialized policies and 1000 episodes per random initialization. RMA beats the performance of all the baselines, with only a slight degradation in performance compared to the Expert.

Discussion

The reward function r_t is the *weighted* sum of

[20, 21, 0.002, 0.02, 0.001, 0.07, 0.002, 1.5, 2.0, 0.8]

1) Forward: $\min(v_x^t, 0.35)$
2) Lateral Movement and Rotation: $-\|v_y^t\|^2 - \|\omega_{\texttt{yaw}}^t\|^2$
3) Work: $-|\boldsymbol{\tau}^T \cdot (\mathbf{q}^t - \mathbf{q}^{t-1})|$
4) Ground Impact: $-\|\mathbf{f}^t - \mathbf{f}^{t-1}\|^2$
5) Smoothness: $-\|\boldsymbol{\tau}^t - \boldsymbol{\tau}^{t-1}\|^2$
6) Action Magnitude: $-\|\mathbf{a}^t\|^2$
7) Joint Speed: $-\|\dot{\mathbf{q}}^t\|^2$
8) Orientation: $-\|\boldsymbol{\theta}_{\texttt{roll, pitch}}^t\|^2$
9) Z Acceleration: $-\|v_z^t\|^2$
10) Foot Slip: $-\|\text{diag}(\mathbf{g}^t) \cdot \mathbf{v_f}^t\|^2$

If naively train the agent with the above reward function, it learns to **stay in place** because of the penalty terms on the movement of the joints.

To prevent this collapse, the training starts with very small penalty coefficients, and then **gradually** increase the strength of these coefficients using a fixed curriculum.

DISCUSSION

@83_f3
For the training curriculum, …

I believe this is an effective method to maintain the reward function without having the collapse. However, I wonder if there are better ways to define the reward function or training curriculum, so that the agent is more "motivated" to move.

@ One Reply of 83_f3
In [1], instead of varying the rewards, the researchers varied the simulation itself to accommodate the robot's current skill level. But from what I remember, this involved hand-designing a measure of "difficulty", which likely took a lot of effort compared to implementing a scaling reward function.

@83_f5
One thought on this work: the reward function seems **highly hand-crafted**. I wonder if the authors tried simpler reward functions and did not see good performance?

In general, it seems like there is no good way to provide general enough rewards (from a human perspective) for these types of task-specific RL problems. I wonder if over time we will develop models that can for example take natural language instructions and learn behaviors that satisfy said instructions. Curiosity-based learning seems to work to some extent, but without any specific reward you might end up with a robot that's really good at doing backflips instead of one that can walk.

[1] Lee J, Hwangbo J, Wellhausen L, et al. Learning quadrupedal locomotion over challenging terrain[J]. Science robotics, 2020, 5(47): eabc5986.

ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
UNIVERSITY OF MICHIGAN

Thanks!