Qualifying Exam: Sim2Real Transfer for Quadruped Robots

Jeremiah Coholich

Contents

Introduction: Motivation

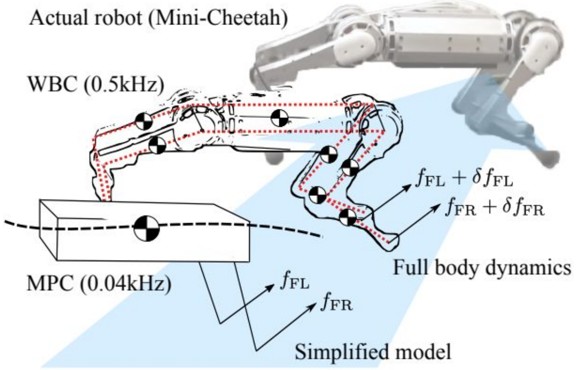
- Wheeled platforms are mostly limited to flat ground
- Legged embodiments can go anywhere humans may go
- This enables embodied intelligence to be applied in a much wider variety of situations

https://news.mit.edu/2018/blind-cheetah-robot-climb-stairs-obstacles-disaster-zones-0705

Lee, Joonho, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. "Learning quadrupedal locomotion over challenging terrain." *Science robotics* 5, no. 47 (2020).

Introduction

- Quadrupedal locomotion is a difficult controls problem
  - Requires expert knowledge of dynamics and task
  - Controllers may be brittle to gains, dynamics model, and contact conditions
  - Hybrid dynamics



Kim, Donghyun, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control." *arXiv preprint arXiv:1909.06586* (2019).

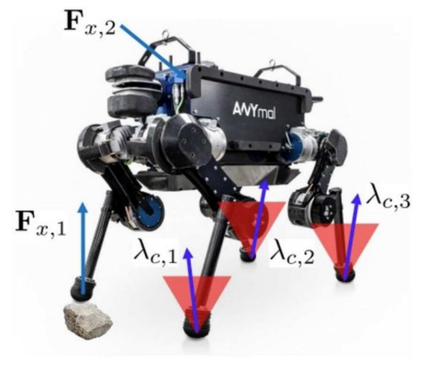


FIGURE 1 | A quadruped robot, ANYmal, with potential external disturbances F_x and contact forces $\boldsymbol{\lambda}_c$ within friction cones during static walking.

Xin, Guiyang, Wouter Wolfslag, Hsiu-Chin Lin, Carlo Tiseo, and Michael Mistry. "An optimization-based locomotion controller for quadruped robots leveraging cartesian impedance control." *Frontiers in Robotics and AI* 7 (2020): 48.

Introduction

- Model-free reinforcement learning is a general way to automatically learn robot locomotion skills
  - Training a robot in simulation is desirable due to sample inefficiency of RL algorithms and dangers of training policies on real robot

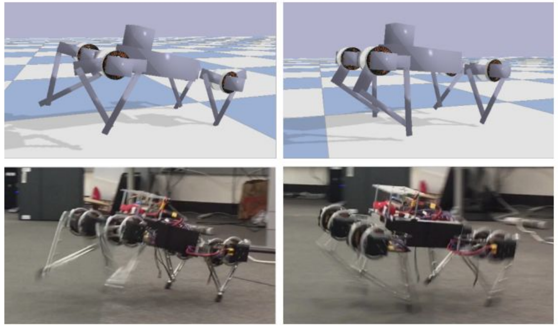


Fig. 1: The simulated and the real Minitaurs learned to gallop using deep reinforcement learning.

Tan, Jie, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. "Sim-to-real: Learning agile locomotion for quadruped robots." *arXiv preprint arXiv:1804.10332* (2018).

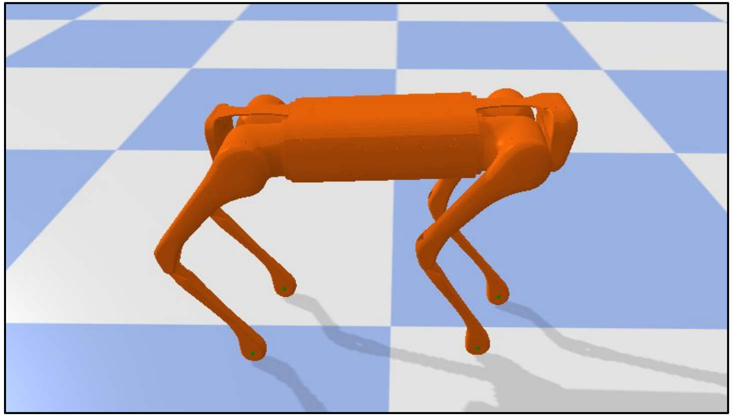
Introduction

Many times in RL, the test environment is the training environment.

Image source:https://syncedreview.com/2020/01/08/slm-lab-new-rl-research-benchmark-software-framework/

Introduction

In sim2real, the test environment is different from the training environment.



Train

Test

Introduction

Formally, the task of quadrupedal locomotion can be represented as a Partially Observable Markov Decision Process (POMDP):

$(S, A, p, r, \Omega, O, \rho_0\ , \gamma)$

$S =$ state space
$A =$ action space
$p : S \times A \mapsto S$ is the environment transition function
$r : S \mapsto R$ is the reward function
$\Omega =$ he observation space
$O =$ is the observation conditional probabilities
ρ_0 = distribution of initial states
$\gamma =$ discount factor

The goal of reinforcement learning is to find the optimal policy π that maximums the discounted sum of rewards.

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} \sum_{t=0}^{T} \gamma^t r\left(\mathbf{s}_t, \mathbf{a}_t\right)$$

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

However, when transferring from sim2real, the POMDP changes unpredictably.

$$O \rightarrow O'$$

$$p \rightarrow p'$$

Introduction

Literature review will cover:

Iscen, Atil, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. "Policies modulating trajectory generators." In Conference on Robot Learning, pp. 916-926. PMLR, 2018.

Yu, Wenhao, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. "Learning fast adaptation with meta strategy optimization." IEEE Robotics and Automation Letters 5, no. 2 (2020): 2950-2957.

Peng, Xue Bin, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. "Learning agile robotic locomotion skills by imitating animals." arXiv preprint arXiv:2004.00784 (2020).

Policies Modulating Trajectory Generators

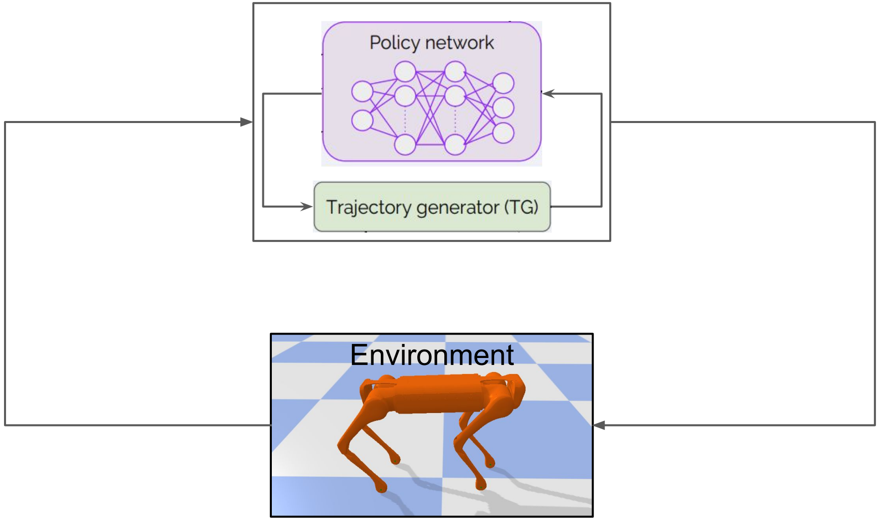
Iscen, Atil, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. "Policies modulating trajectory generators." In Conference on Robot Learning, pp. 916-926. PMLR, 2018.

Literature Review: Policies Modulating Trajectory Generators

Key Ideas for sim2real transfer: Domain randomization and small observation space

Observation:
- IMU (pitch, roll, pitch velocity, roll velocity)
- Speed command

Reward:
- Track speed command



Action:
- Joint position targets

Iscen, Atil, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. "Policies modulating trajectory generators." In Conference on Robot Learning, pp. 916-926. PMLR, 2018.

Literature Review: Policies Modulating Trajectory Generators

TABLE I
RANDOMIZED PARAMETERS AND THEIR RANGE USED IN TRAINING.

parameter	lower bound	upper bound
mass	60%	160%
motor friction	0.0Nm	0.2Nm
inertia	25%	200%
motor strength	50%	150%
latency	0ms	80ms
battery voltage	10V	18V
contact friction	0.2	1.25
joint friction	0.0Nm	0.2Nm

Yu, Wenhao, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. "Learning fast adaptation with meta strategy optimization." *IEEE Robotics and Automation Letters* 5, no. 2 (2020): 2950-2957.

Akkaya, Ilge, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino et al. "Solving rubik's cube with a robot hand." *arXiv preprint arXiv:1910.07113* (2019).

Iscen, Atil, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. "Policies modulating trajectory generators." In Conference on Robot Learning, pp. 916-926. PMLR, 2018.

Literature Review: Policies Modulating Trajectory Generators

The learned policy transfers directly to the real world.

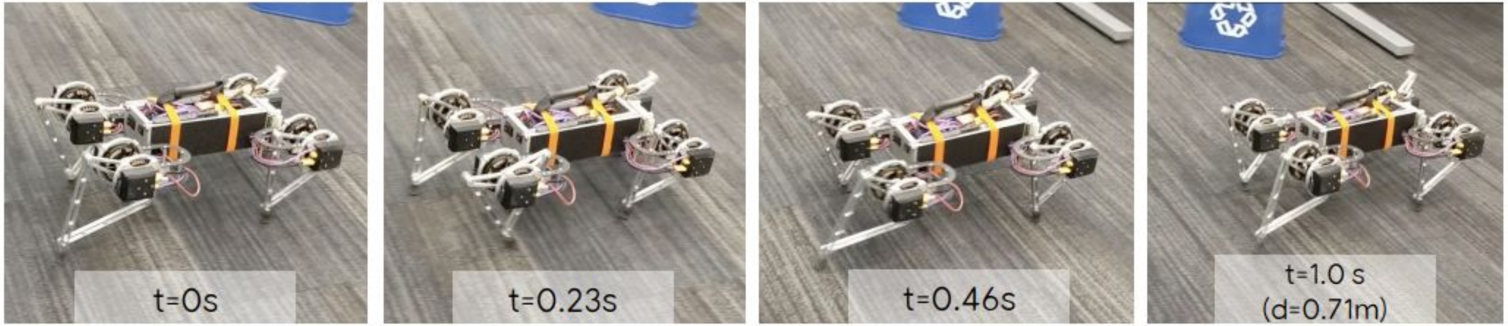


Figure 9: Minitaur robot walking using the learned controller.

Iscen, Atil, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. "Policies modulating trajectory generators." In Conference on Robot Learning, pp. 916-926. PMLR, 2018.

Literature Review: Policies Modulating Trajectory Generators

The sim2real strategy relies on robustness of the policy

- Small observation space plus randomization prevents the policy from overfitting to training environment
- However, optimality is sacrificed for robustness

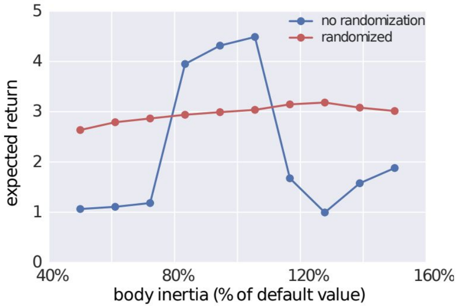


Fig. 7: Performance comparison of controllers that are trained with (red) and without (blue) randomization and tested with different body inertia.

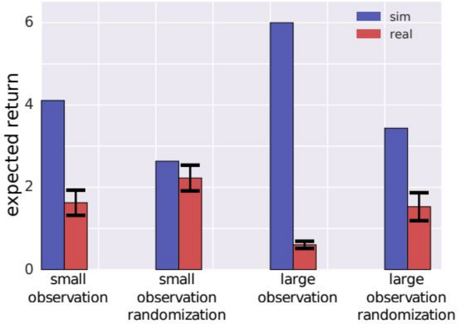


Fig. 9: Comparison of controllers trained with different observation spaces and randomization. The blue and red bars are the performance in simulation and in the real world respectively. Error bars indicate one standard error.

Figures from:
Tan, Jie, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. "Sim-to-real: Learning agile locomotion for quadruped robots." *arXiv preprint arXiv:1804.10332* (2018).

Learning Fast Adaptation with Meta Strategy Optimization

Yu, Wenhao, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. "Learning fast adaptation with meta strategy optimization." IEEE Robotics and Automation Letters 5, no. 2 (2020): 2950-2957.

Literature Review: Learning Fast Adaptation with Meta Strategy Optimization

Main idea

- Condition policy on a latent space which adapts to environment parameters.
- Collect samples from real robot to train latent space to adapt to real world.

Task: Forward locomotion
Observation: joint states, IMU
Action Space: joint positions
Reward: forward speed

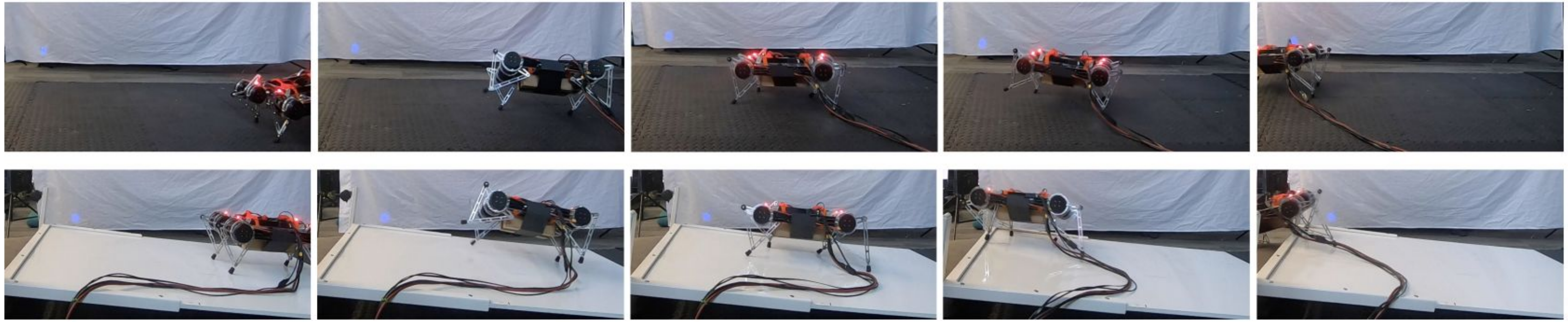
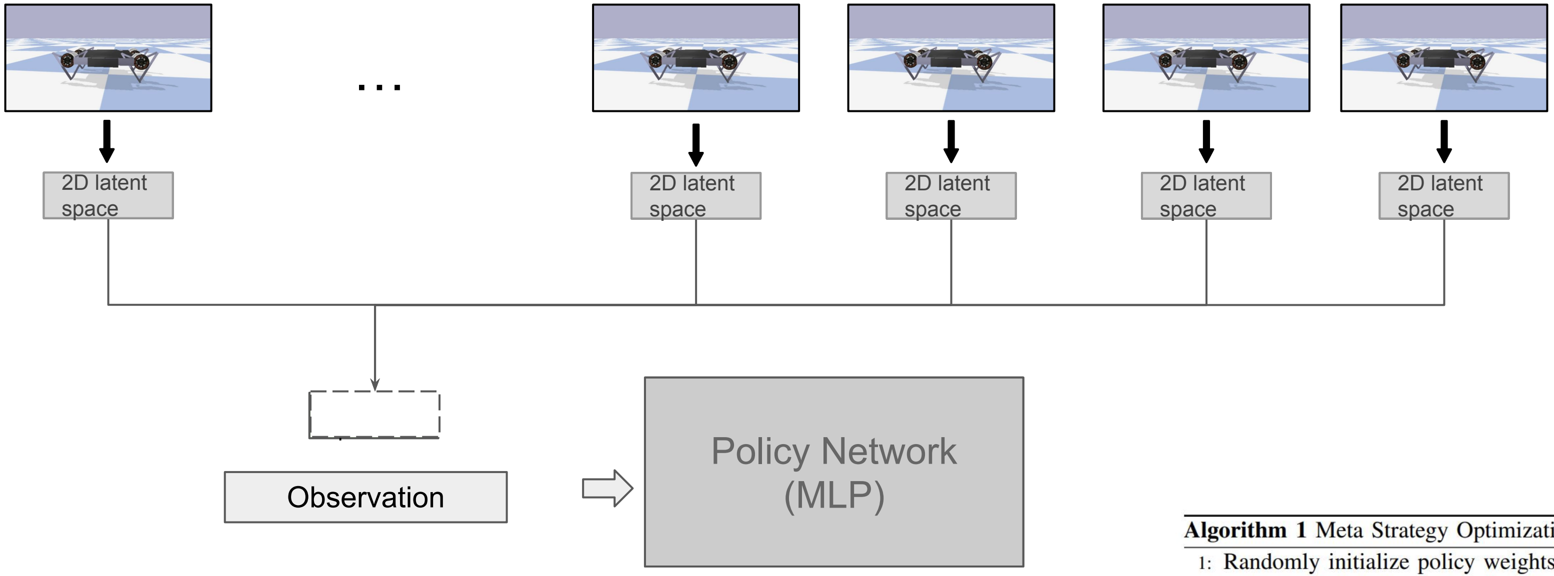


Fig. 6. Policy trained by MSO adapts to new tasks: front right leg weakened (top), walking up a slope (bottom).

Yu, Wenhao, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. "Learning fast adaptation with meta strategy optimization." IEEE Robotics and Automation Letters 5, no. 2 (2020): 2950-2957.

Literature Review: Learning fast adaptation with meta strategy optimization

Create n environments with physics parameters $\mu_1, \mu_2, \ldots, \mu_n$



Latent space and policy are trained with Augmented Random Search (ARS). Training process has no explicit knowledge of μ.

Repeat, creating new environments each iteration

Algorithm 1 Meta Strategy Optimization

```
Randomly initialize policy weights θ_1.
for t = 1 : k do
    Sample n tasks {μ_i | i = 1, ..., n}.
    For each μ_i, solve Eq. 5 with θ_t and obtain c_{μ_i,t}.
    for j = 1 : h do
        Randomly sample a pair of (c_{μ,t}, μ).
        Collect rollouts with p_μ and π_{θ_t}(s, c_{μ,t}).
        Obtain θ_{t+1} by solving Equation 6.
return π_{θ_k}
```

Literature Review: Learning Fast Adaptation with Meta Strategy Optimization

- Adaptation is successful with < 4000 samples gathered on real robot (about 75 seconds)
- Adapts to a wide range of tasks not encountered during training (walking up slope, weakened motor, wide randomization range)

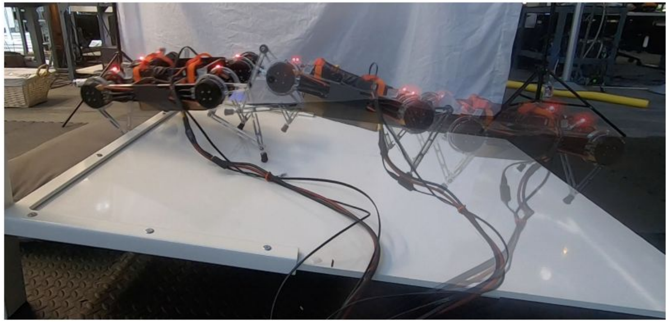


Fig. 1. Policies trained using our method adapts to sloped surface on the real quadruped robot in 15 episodes. During training in simulation, it has only seen flat ground.

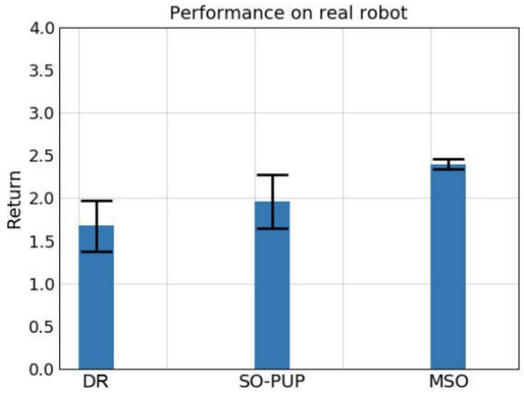


Fig. 3. Sim-to-real performance comparison on the Minitaur robot (corresponding to Task 1: Sim-to-real transfer as described in V-B). Error bar denotes on standard deviation.

Yu, Wenhao, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. "Learning fast adaptation with meta strategy optimization." IEEE Robotics and Automation Letters 5, no. 2 (2020): 2950-2957.

Literature Review: Learning Fast Adaptation with Meta Strategy Optimization

Drawbacks

- Samples must be collected on the real robot (no free lunch)
- Training on the real robot requires a motion capture system in order to obtain robot state to give rewards
- New samples must be collected whenever the environment changes

Yu, Wenhao, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. "Learning fast adaptation with meta strategy optimization." IEEE Robotics and Automation Letters 5, no. 2 (2020): 2950-2957.

Learning Agile Robot Locomotion Skills by Imitating Animals

Peng, Xue Bin, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. "Learning agile robotic locomotion skills by imitating animals." arXiv preprint arXiv:2004.00784 (2020).

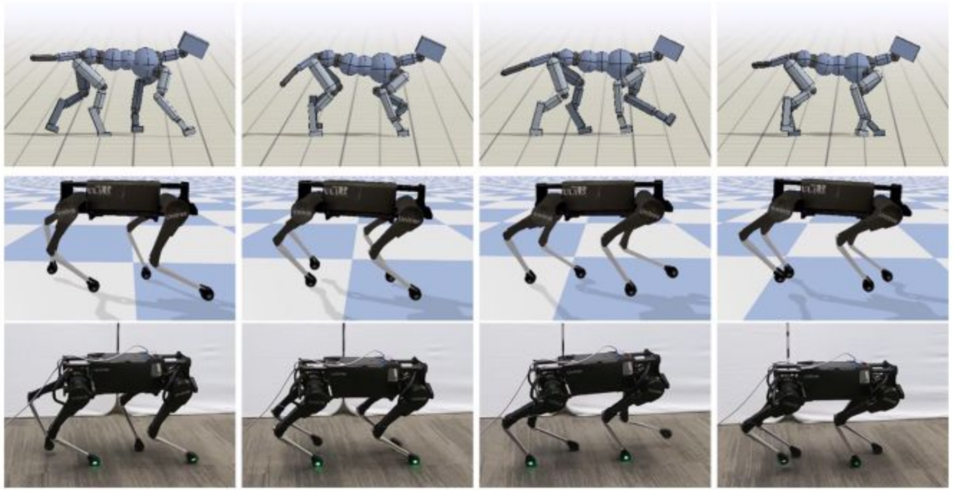
Literature Review: Learning Agile Robotic Locomotion Skills by Imitating Animals

Task: Imitate motion capture clips from animals

Observation: previous action, robot joint states, goal joint states

Action Space: joint positions

Reward: track reference trajectory

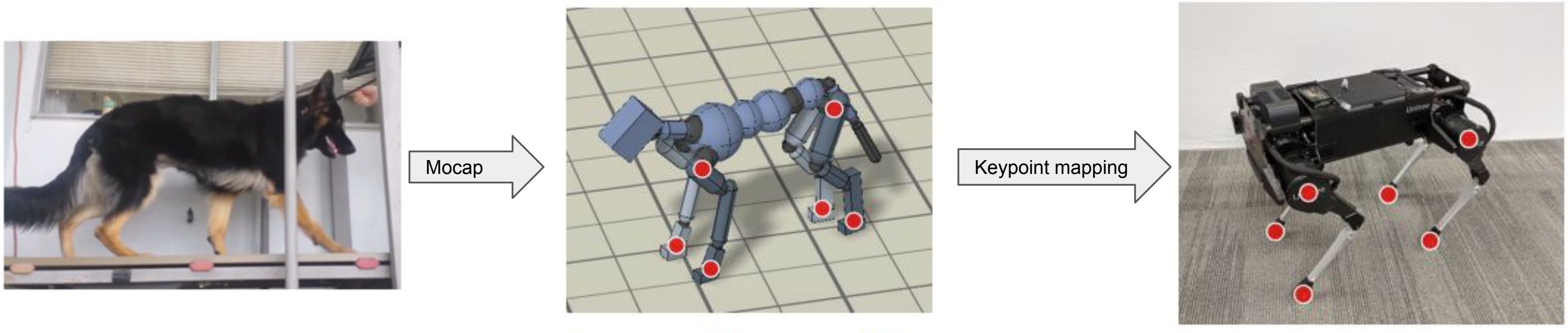


Peng, Xue Bin, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. "Learning agile robotic locomotion skills by imitating animals." arXiv preprint arXiv:2004.00784 (2020).

Literature Review: Learning Agile Robotic Locomotion Skills by Imitating Animals

The pipeline consists of three steps:

1) Motion Targeting



2) Motion Imitation
    - Robot is trained to imitate the reference behavior in a PyBullet simulation
3) Domain Adaptation via a latent space method

Peng, Xue Bin, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. "Learning agile robotic locomotion skills by imitating animals." arXiv preprint arXiv:2004.00784 (2020).

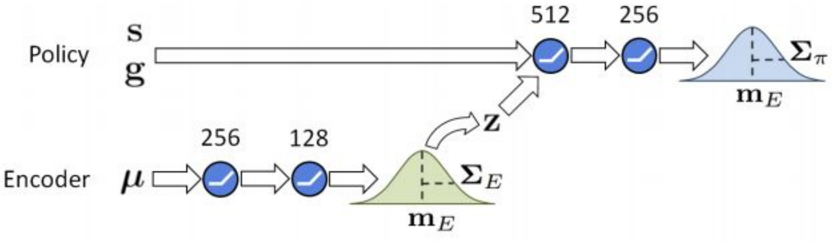
Literature Review: Learning Agile Robotic Locomotion Skills by Imitating Animals

Training

- Latent space is sampled from stochastic encoder, which has access to simulation physics parameters (μ)
- Trained with Proximal Policy Optimization (PPO)
- Uses information bottleneck loss term

Testing

- Latent space is learned via Advantage Weighted Regression (AWR) through collecting samples on the real robot



Policy Network Architecture

$$\arg\max_{\pi, E} \; \mathbb{E}_{\boldsymbol{\mu}\sim p(\boldsymbol{\mu})} \mathbb{E}_{\mathbf{z}\sim E(\mathbf{z}|\boldsymbol{\mu})} \mathbb{E}_{\tau\sim p(\tau|\pi,\boldsymbol{\mu},\mathbf{z})} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right] - \beta \, \mathbb{E}_{\boldsymbol{\mu}\sim p(\boldsymbol{\mu})} \left[\mathrm{D_{KL}} \left[E(\cdot|\boldsymbol{\mu}) || \rho(\cdot) \right] \right], \quad (14)$$

Policy Network Optimization Problem (Next Slide)

Peng, Xue Bin, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. "Learning agile robotic locomotion skills by imitating animals." arXiv preprint arXiv:2004.00784 (2020).

Literature Review: Learning Agile Robotic Locomotion Skills by Imitating Animals

$$\arg\max_{\pi, E} \mathbb{E}_{\boldsymbol{\mu} \sim p(\boldsymbol{\mu})} \mathbb{E}_{\mathbf{z} \sim E(\mathbf{z}|\boldsymbol{\mu})} \mathbb{E}_{\tau \sim p(\tau|\pi,\boldsymbol{\mu},\mathbf{z})} \boxed{\left[\sum_{t=0}^{T-1} \gamma^t r_t \right]}$$
$$- \beta \boxed{\mathbb{E}_{\boldsymbol{\mu} \sim p(\boldsymbol{\mu})} \left[\mathrm{D_{KL}} \left[E(\cdot|\boldsymbol{\mu}) || \rho(\cdot) \right] \right]},$$

Discounted sum of rewards

Information bottleneck (IB) term, derived from minimizing mutual information between μ and z. (Appendix)

μ : Simulation physics parameters
$\mathbf{z}$: Latent space
τ : Trajectory of states
π : Policy
γ : Discount factor
r_t : Reward at timestep t
β : Hyperparameter
$\rho(\cdot)$: Variational prior, chosen to be $\mathcal{N}(0, 1)$
$E(\cdot|\mu)$: Encoder

Literature Review: Learning Agile Robotic Locomotion Skills by Imitating Animals

Drawbacks

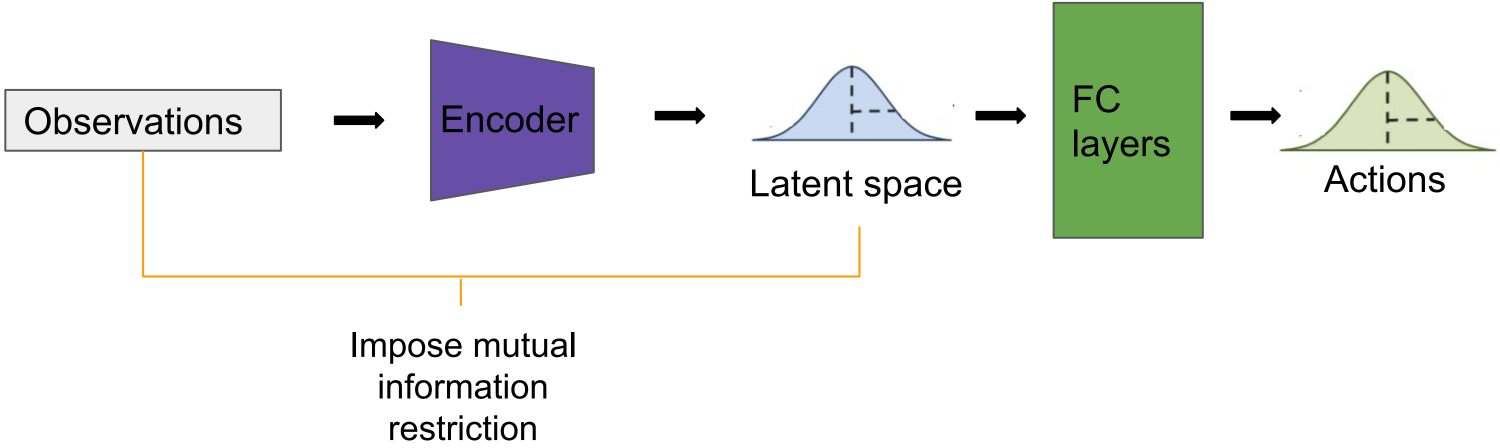
- This approach is not able to learn a general controller for the quadruped
- Collecting mocap data from animals is not a scalable data collection pipeline
    - Adding artist-generated animations to the dataset is perhaps evidence of this

Conclusion: Comparison

- *Policies Modulating Trajectory Generators* (Method A) primarily relies on robustness, which is desirable for its simplicity and works when the sim2real gap is small.
  - Requires no real robot training.
  - Potentially sacrifices performance.
- *Learning Agile Robotic Locomotion Skills by Imitating Animals* (Method C) is the most powerful formulation, since it enables a continuously-tunable tradeoff between robustness and adaptability via the hyperparameter β.
  - *Learning Fast with Meta Strategy Optimization* (Method B) controls this tradeoff through the size of the latent space. However, increasing the latent space size greatly increases required computation, since the latent space is found through random search.
  - Method C therefore scales better to more complex tasks with larger sim2real gaps.

Conclusion: Potential Future Work

- Pass observations through an information bottleneck to improve generalization performance
- Instead of limiting the observation space to only IMU data like Method A, learn what to exclude
- Continuously tune the amount of exclusion
- This method is more general



Appendix

Appendix: Derivation of Stochastic Encoder Regularizing loss term from Mutual Information Constraint

The authors aim to minimize the mutual information between the input (simulation dynamics parameters), $\mathbf{M}$, and the latent space (encoder output) $\mathbf{Z}$. Mutual information is given by:

$$\mathbf{I}(\mathbf{M}, \mathbf{Z}) = \int \int p(m, z) \log \frac{p(m, z)}{p(m)p(z)} \, dm \, dz = \int \int p(m, z) \log \frac{p(z|m)}{p(z)} \, dm \, dz \tag{1}$$

$$= \int \int p(m, z) \log p(z|m) \, dm \, dz - \int \int p(m, z) \log p(z) \, dm \, dz \tag{2}$$

However, this is difficult to compute, so $p(z)$ is approximated with $q(z)$. Using the fact that KL-divergence is always positive:

$$\mathrm{D_{KL}}(p(z) \parallel q(z)) = \int p(z) \log \frac{p(z)}{q(z)} \, dz \geq 0 \Rightarrow \int p(z) \log p(z) \, dz \geq \int p(z) \log q(z) \, dz \tag{3}$$

$$\Rightarrow \int \int p(m, z) \log p(z) \, dz \, dm \geq \int \int p(m, z) \log q(z) \, dz \, dm \tag{4}$$

Appendix: Derivation of Stochastic Encoder Regularizing loss term from Mutual Information Constraint

Combining (2) and (4) yields:

$$\mathbf{I}(\mathbf{M},\mathbf{Z}) \le \int\int p(m,z)\,\log p(z|m)\,dm\,dz - \int\int p(m,z)\,\log q(z)\,dm\,dz \tag{5}$$

$$= \int\int p(m)p(z|m)\,\log\frac{p(z|m)}{q(z)}\,dm\,dz \tag{6}$$

$$= \mathbb{E}_{\mathrm{M}}\left[\int p(z|m)\,\log\frac{p(z|m)}{q(z)}\,dz\right] \tag{7}$$

$$= \mathbb{E}_{\mathrm{M}}\left[\mathrm{D}_{\mathrm{KL}}\left(p(z|m)\,\|\,q(z)\right)\right] \tag{8}$$

$p(z|m)$ is the encoder network. The authors chose the output of the encoder and variational prior $q(z)$ to both be Gaussians so that an analytical KL-diverengce can be computed and optimized with gradient descent.

Back

Reference: Alemi, Alexander A., Ian Fischer, Joshua V. Dillon, and Kevin Murphy. "Deep variational information bottleneck." *arXiv preprint arXiv:1612.00410* (2016).

Appendix: Advantage Weighted Regression

- Off-policy algorithm

Algorithm 1 Adaptation with Advantage-Weighted Regression

$\pi \leftarrow$ trained policy
$\omega_0 \leftarrow \mathcal{N}(0, I)$
$\mathcal{D} \leftarrow \emptyset$
for iteration $k = 0, ..., k_{\max} - 1$ **do**
    $\mathbf{z}_k \leftarrow$ sampled encoding from $\omega_k(\mathbf{z})$
    Rollout an episode with π conditioned $\mathbf{z}_k$ and record the return $\mathcal{R}_k$
    Store $(\mathbf{z}_k, \mathcal{R}_k)$ in $\mathcal{D}$
    $\bar{v} \leftarrow \frac{1}{k} \sum_{i=1}^{k} \mathcal{R}_i$
    $\omega_{k+1} \leftarrow \arg\max_{\omega} \sum_{i=1}^{k} \left[\log \omega(\mathbf{z}_i) \exp\left(\frac{1}{\alpha}\left(\mathcal{R}_i - \bar{v}\right)\right)\right]$
end for

Peng, Xue Bin, Aviral Kumar, Grace Zhang, and Sergey Levine. "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning." *arXiv preprint arXiv:1910.00177* (2019). (https://arxiv.org/pdf/1910.00177.pdf)

Appendix: Augmented Random Search

Algorithm 2 Augmented Random Search (ARS): four versions **V1**, **V1-t**, **V2** and **V2-t**

Hyperparameters: step-size α, number of directions sampled per iteration N, standard deviation of the exploration noise ν, number of top-performing directions to use b ($b < N$ is allowed only for **V1-t** and **V2-t**)

Initialize: $M_0 = \mathbf{0} \in \mathbb{R}^{p \times n}$, $\mu_0 = \mathbf{0} \in \mathbb{R}^n$, and $\Sigma_0 = \mathbf{I}_n \in \mathbb{R}^{n \times n}$, $j = 0$.

while ending condition not satisfied **do**

  Sample $\delta_1, \delta_2, \ldots, \delta_N$ in $\mathbb{R}^{p \times n}$ with i.i.d. standard normal entries.

  Collect $2N$ rollouts of horizon H and their corresponding rewards using the $2N$ policies

$$\textbf{V2:}\quad \begin{cases} \pi_{j,k,+}(x) = (M_j + \nu\delta_k)\,\mathrm{diag}\left(\Sigma_j\right)^{-1/2}(x - \mu_j) \\ \pi_{j,k,-}(x) = (M_j - \nu\delta_k)\,\mathrm{diag}(\Sigma_j)^{-1/2}(x - \mu_j) \end{cases}$$

  for $k \in \{1, 2, \ldots, N\}$.

  Sort the directions δ_k by $\max\{r(\pi_{j,k,+}), r(\pi_{j,k,-})\}$, denote by $\delta_{(k)}$ the k-th largest direction, and by $\pi_{j,(k),+}$ and $\pi_{j,(k),-}$ the corresponding policies.

  Make the update step:

$$M_{j+1} = M_j + \tfrac{\alpha}{b\sigma_R} \sum_{k=1}^{b} \left[r(\pi_{j,(k),+}) - r(\pi_{j,(k),-})\right] \delta_{(k)},$$

  where σ_R is the standard deviation of the $2b$ rewards used in the update step.

  V2 : Set μ_{j+1}, Σ_{j+1} to be the mean and covariance of the $2NH(j+1)$ states encountered from the start of training.[2]

  $j \leftarrow j + 1$

end while

.

Mania, Horia, Aurelia Guy, and Benjamin Recht. "Simple random search provides a competitive approach to reinforcement learning." *arXiv preprint arXiv:1803.07055* (2018). (https://arxiv.org/pdf/1803.07055.pdf)

Appendix: Proximal Policy Optimization

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

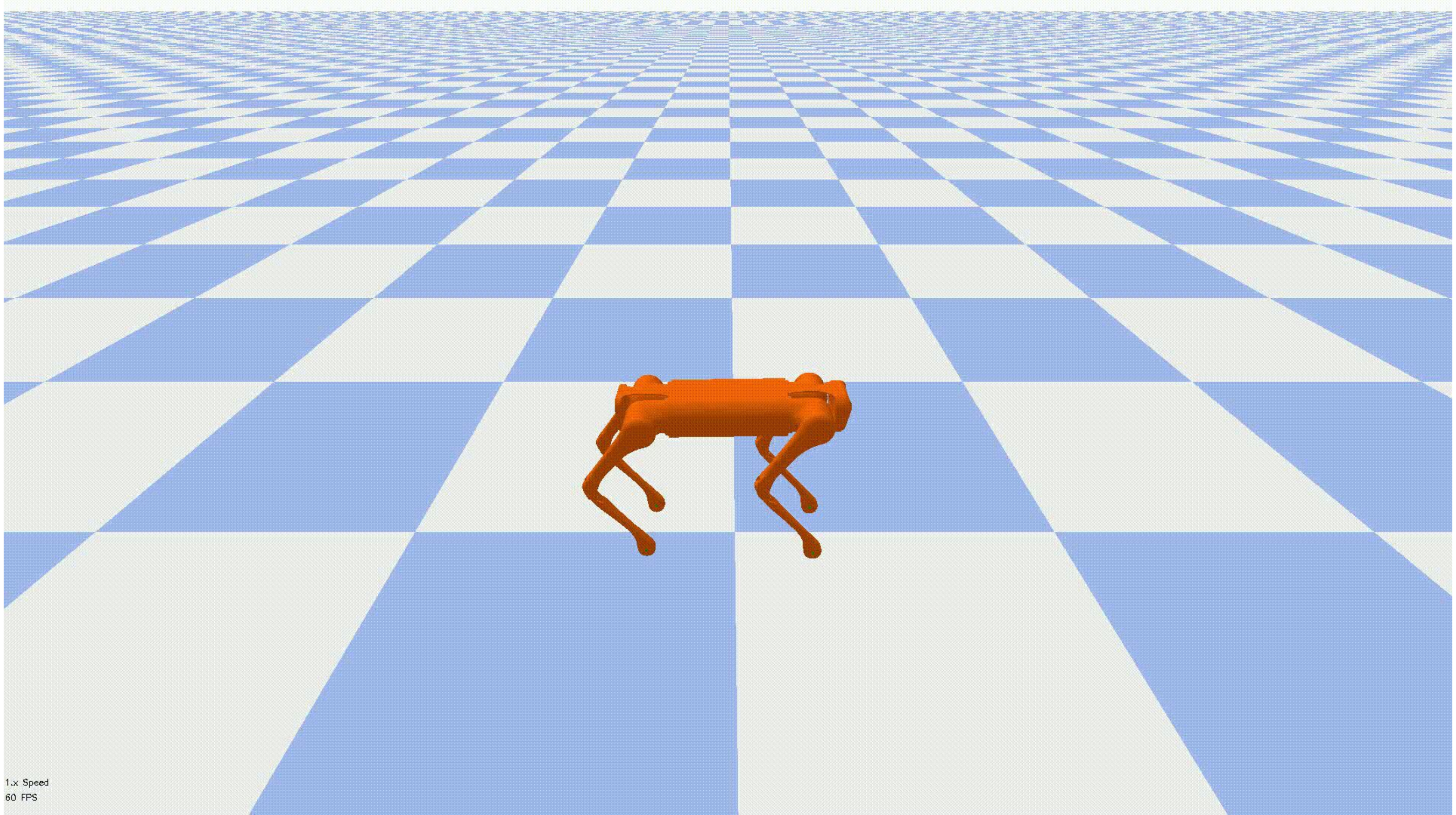
Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017). (https://arxiv.org/pdf/1707.06347.pdf)

Appendix: Vanilla Policy Gradient

$$
\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}[R(\tau)] & \\
&= \nabla_\theta \int_\tau P(\tau|\theta)R(\tau) & \text{Expand expectation} \\
&= \int_\tau \nabla_\theta P(\tau|\theta)R(\tau) & \text{Bring gradient under integral} \\
&= \int_\tau P(\tau|\theta)\nabla_\theta \log P(\tau|\theta)R(\tau) & \text{Log-derivative trick} \\
&= \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}[\nabla_\theta \log P(\tau|\theta)R(\tau)] & \text{Return to expectation form} \\
\therefore\ \nabla_\theta J(\pi_\theta) &= \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau)\right] & \text{Expression for grad-log-prob}
\end{aligned}
$$

https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html

Appendix: Policies Modulating Trajectory Generators

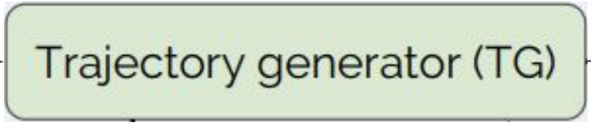


Appendix: Policies Modulating Trajectory Generators

Main Idea: Learning from scratch is time consuming and hard, use prior knowledge in the form of a trajectory (i.e. gait) generator

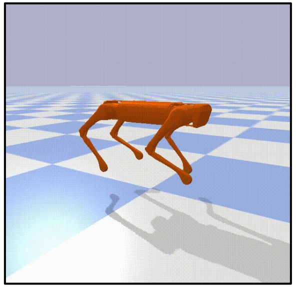
Inputs

- Frequency
- Step length
- Base height



Output

- Open-loop gait pattern



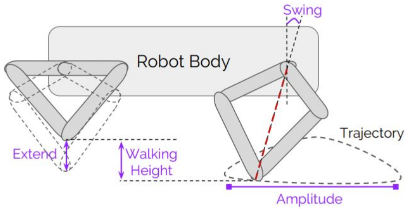


Figure 4: Illustration of robot leg trajectories generated by the TG.

Iscen, Atil, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. "Policies modulating trajectory generators." In Conference on Robot Learning, pp. 916-926. PMLR, 2018.

Appendix: Policies Modulating Trajectory Generators

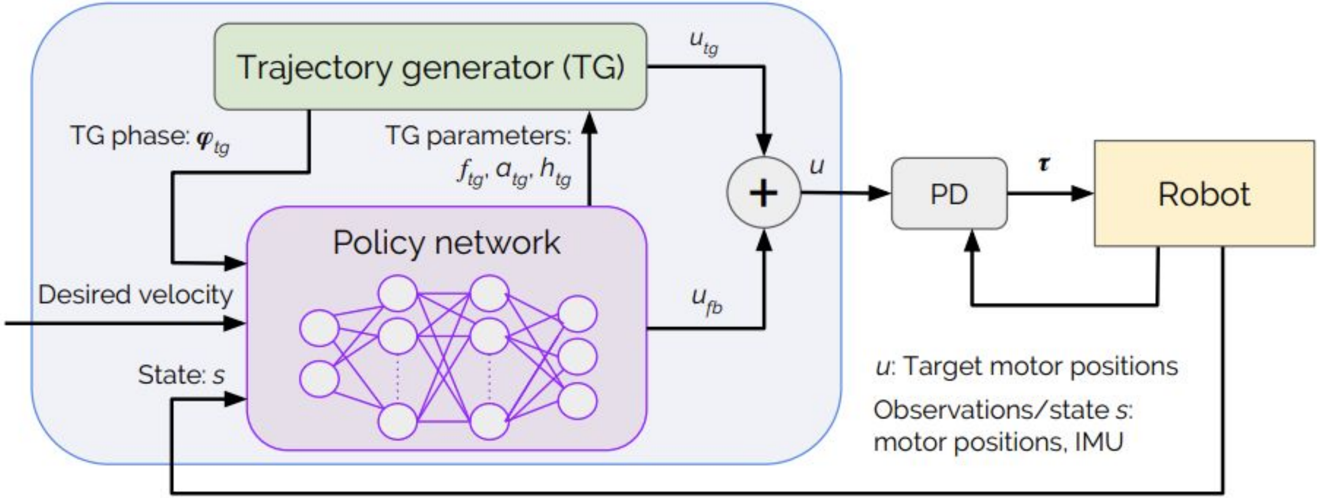


Figure 5: Adaptation of PMTG to the quadruped locomotion problem.

Iscen, Atil, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. "Policies modulating trajectory generators." In Conference on Robot Learning, pp. 916-926. PMLR, 2018.