Learning a Contact-Adaptive Controller for Robust, Efficient Legged Locomotion

Hansol Kang, Robotics Innovatory, SKKU

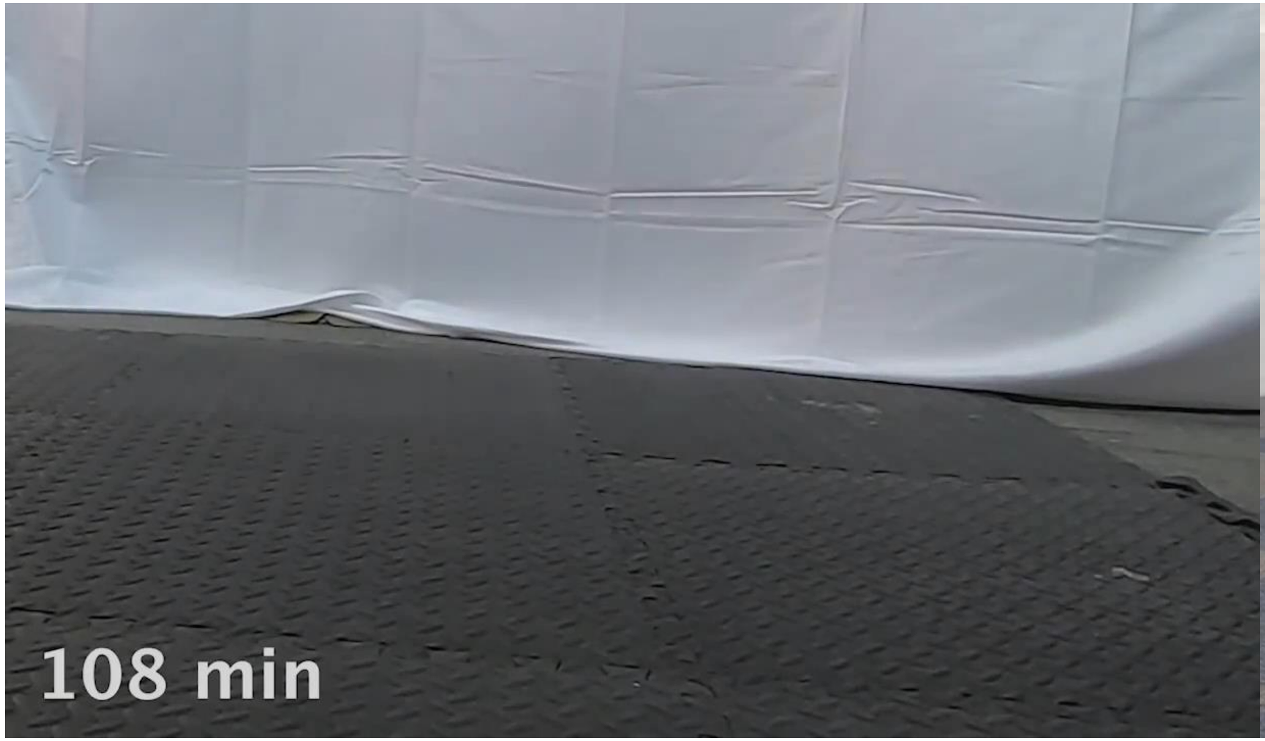
http://mecha.skku.ac.kr

Contents

- Introduction
- Paper Review
- Implementation

Introduction

- Robots controlled through RL



Learning to Walk via Deep Reinforcement Learning. Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, Sergey Levine. Robotics: Science and Systems (RSS). 2019.
Multi-expert learning of adaptive legged locomotion, Chuanyu Yang, Kai Yuan, Qiuguo Zhu, Wanming Yu, Zhibin LiScience Robotics, 2020

Paper Review

- Learning a Contact-Adaptive Controller for Robust, Efficient Legged Locomotion

arXiv:2009.10019v4 [cs.RO] 23 Nov 2020

Learning a Contact-Adaptive Controller for Robust, Efficient Legged Locomotion

Xingye Da[*], Zhaoming Xie[*†], David Hoeller[*], Byron Boots[*‡], Animashree Anandkumar[*§], Yuke Zhu[*♯], Buck Babich[*], Animesh Garg[*⋆]

Abstract: We present a hierarchical framework that combines model-based control and reinforcement learning (RL) to synthesize robust controllers for a quadruped (the Unitree Laikago). The system consists of a high-level controller that learns to choose from a set of primitives in response to changes in the environment and a low-level controller that utilizes an established control method to robustly execute the primitives. Our framework learns a controller that can adapt to challenging environmental changes on the fly, including novel scenarios not seen during training. The learned controller is up to 85 percent more energy efficient and is more robust compared to baseline methods. We also deploy the controller on a physical robot without any randomization or adaptation scheme.

Keywords: Legged Locomotion, Hierarchical Control, Reinforcement Learning

1 Introduction

Quadruped locomotion is often characterized in terms of *gaits* (walking, trotting, galloping, bounding, etc.) that have been well-studied in animals [1] and reproduced on robots [2, 3]. A gait is a periodic contact sequence that defines a specific contact timing for each foot. Controllers designed for these gaits have demonstrated robust behaviors on flat ground and rough terrain locomotion. However, it is rarer to find controllers that can change gaits or contact sequences to adapt to environmental changes. An adaptive gait can reduce energy usage by removing unnecessary movement, as suggested in horse studies [1]. It is also required for completing more challenging scenarios such as riding a skateboard or recovery from leg slipping, as shown in Figure 1 (a, b).

In most model-based and learning-based control designs, the contact sequence is fixed or predefined [2, 3, 4, 5, 6, 7, 8]. Dynamic adaptation of the contact sequence is challenging because of the hybrid nature of legged locomotion dynamics as well as the inherent instability of such systems. While it is possible to generate adaptive contact schemes via trajectory optimization [9, 10, 11], such approaches are generally too compute-intensive for real-time use.

Here we present a hierarchical control framework for quadrupedal locomotion that learns to adaptively change contact sequences in real-time. A high-level controller is trained with reinforcement learning (RL) to specify the contact configuration of the feet, which is then taken as input by a low-level controller to generate ground reaction forces via quadratic programming (QP). At inference time, the high-level controller needs only evaluate a small multi-layer neural network, avoiding the use of an expensive model predictive control (MPC) strategy that might otherwise be required to optimize for long-term performance. The low-level controller provides high-bandwidth feedback to track base and foot positions and also helps ensure that learning is sample-efficient. The framework produces a controller that is up to 85 percent more energy efficient and also more robust than baseline approaches.

We train our controller with a simulated Unitree Laikago [12] on a split-belt treadmill, as shown in Figure 1 (c). The two belts can adjust speed independently, and we change the robot orientation to increase variation. In addition to comparing energy use and robustness to the baselines, we also demonstrate zero-shot transferability by testing the controller in novel situations such as one where a foot encounters a slippery surface (e.g., with zero friction), which we call the "banana peel" test.

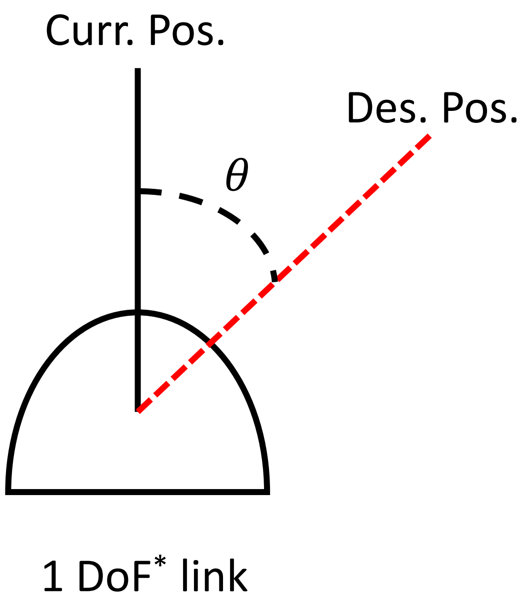
[*]NVIDIA, [†]Univ. of British Columbia, [‡]Univ. of Washington, [§]Caltech, [♯]UT Austin, [⋆]Univ. of Toronto, Vector Institute. Work done at NVIDIA. Correspondence: xda@nvidia.com, zxie47@cs.ubc.ca

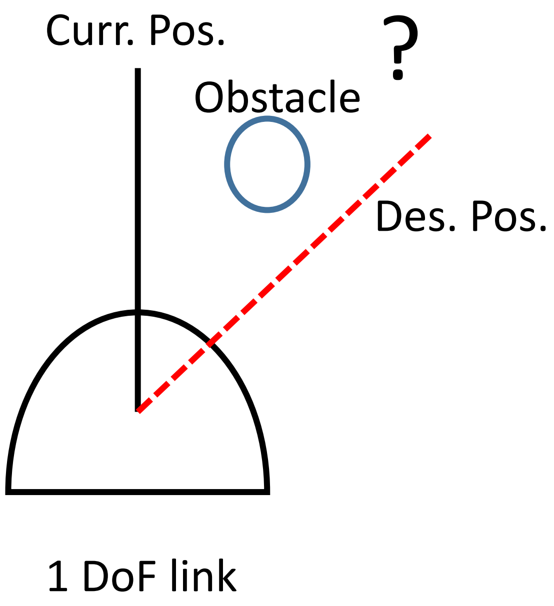
4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

- 2020 Conference on Robot Learning(CoRL)
- 2020.11. arXiv
- Xingye Da[1], Zhaoming Xie[1,2], David Hoeller[1], Byron Boots[1,3], Anima Anandkumar[1,4], Yuke Zhu[1,5], Buck Babich[1], Animech Garg[1,6]
- [1]NVIDIA, [2]Univ. of British Columbia, [3]Univ. of Washington, [4]Calthec, [5]UT Austin, [6]Univ. of Toronto
- Introduce a hierarchical control structure that combines model-based control design and model-free reinforcement learning for legged locomotion.

Paper Review

- How to control a robot?





* Degree of Freedom

Paper Review

Quadratic Programming(QP)

QP is an optimization tool

$$minimize\ 2x_1^2 + x_1x_2 + x_2^2 + x_1 + x_2$$

$$\begin{aligned} subject\ to\ x_1 + x_2 &= 1 \\ 0 \le x_1 &\le 0.7 \\ 0 \le x_2 &\le 0.7 \end{aligned}$$

$$minimize\ \frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathbf{T}}\mathbf{x}$$

$$subject\ to\ \mathbf{A}\mathbf{x} \le \mathbf{b}$$

$$\text{minimize} \quad \frac{1}{2}x^T \begin{bmatrix} 4 & 1 \\ 1 & 2 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix}^T x$$

$$\text{subject to} \quad \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \le \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} x \le \begin{bmatrix} 1 \\ 0.7 \\ 0.7 \end{bmatrix}$$

Paper Review

- How to use QP for quadruped robot control?

$$\underbrace{\begin{bmatrix} \boldsymbol{I} & \cdots & \boldsymbol{I} \\ [\boldsymbol{p}_{com,1}\times] & \cdots & [\boldsymbol{p}_{com,c}\times] \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \boldsymbol{f}_1 \\ \cdots \\ \boldsymbol{f}_c \end{bmatrix}}_{\boldsymbol{f}} = \underbrace{\begin{bmatrix} m(\ddot{\boldsymbol{x}}_{com}^{d} + \boldsymbol{g}) \\ \boldsymbol{I}_g \dot{\boldsymbol{\omega}}_b^d \end{bmatrix}}_{\boldsymbol{b}}$$

$$\boldsymbol{f}^d = \arg\min_{\boldsymbol{f}\in\mathbb{R}^k} (\boldsymbol{A}\boldsymbol{f} - \boldsymbol{b})^T \boldsymbol{S} (\boldsymbol{A}\boldsymbol{f} - \boldsymbol{b}) + \alpha \boldsymbol{f}^T \boldsymbol{W} \boldsymbol{f}$$

$$s.t.\ \underline{\boldsymbol{d}} < \boldsymbol{C}\boldsymbol{f} < \overline{\boldsymbol{d}}$$

$$\boldsymbol{S} \in \mathbb{R}^{6\times 6}\ and\ \boldsymbol{W} \in \mathbb{R}^{k\times k}\ are\ positive-definite\ weight\ matrices$$

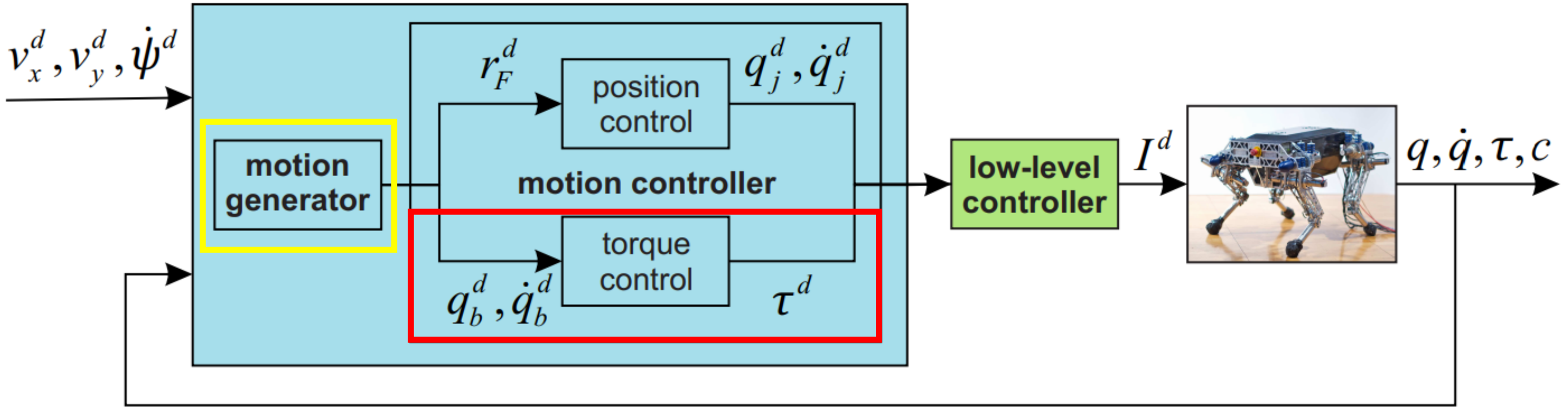
$$\underline{\boldsymbol{d}}, \overline{\boldsymbol{d}} \in \mathbb{R}^p, \boldsymbol{C} \in \mathbb{R}^{p\times k}$$

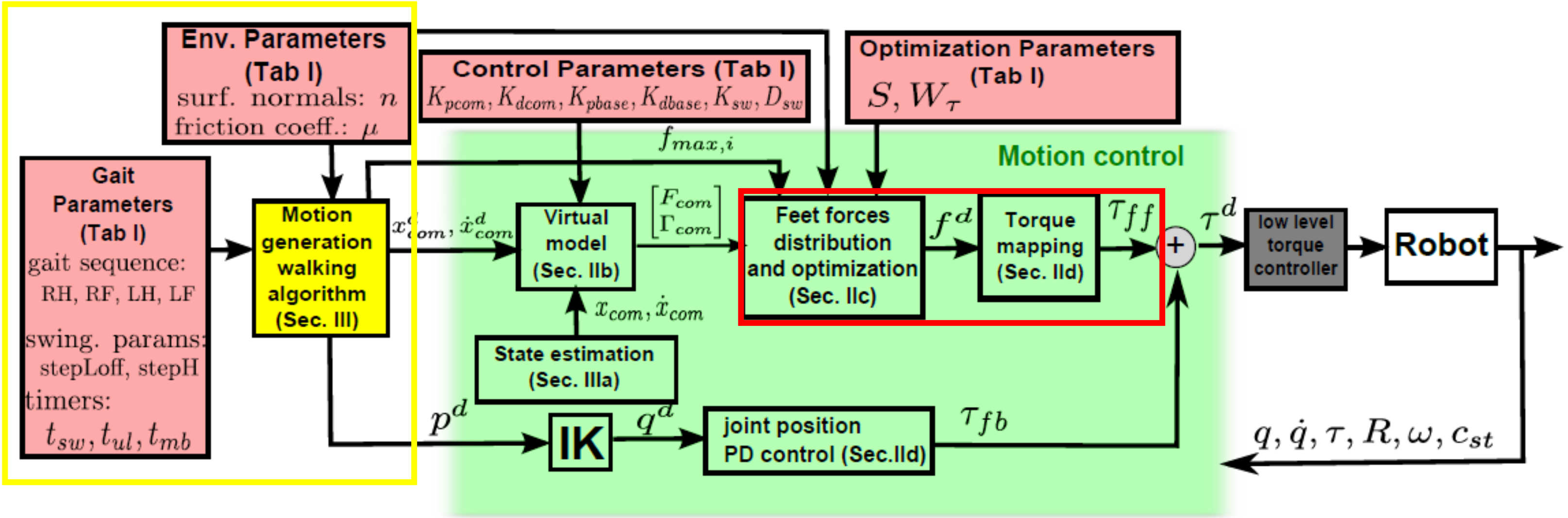
$$k = 3 \times Num.of\ contact, p = Num.of\ inequality\ const.$$

$$\alpha \in \mathbb{R}\ weighs\ the\ secondary\ objective.$$

Paper Review

- How to use QP for quadruped robot control?

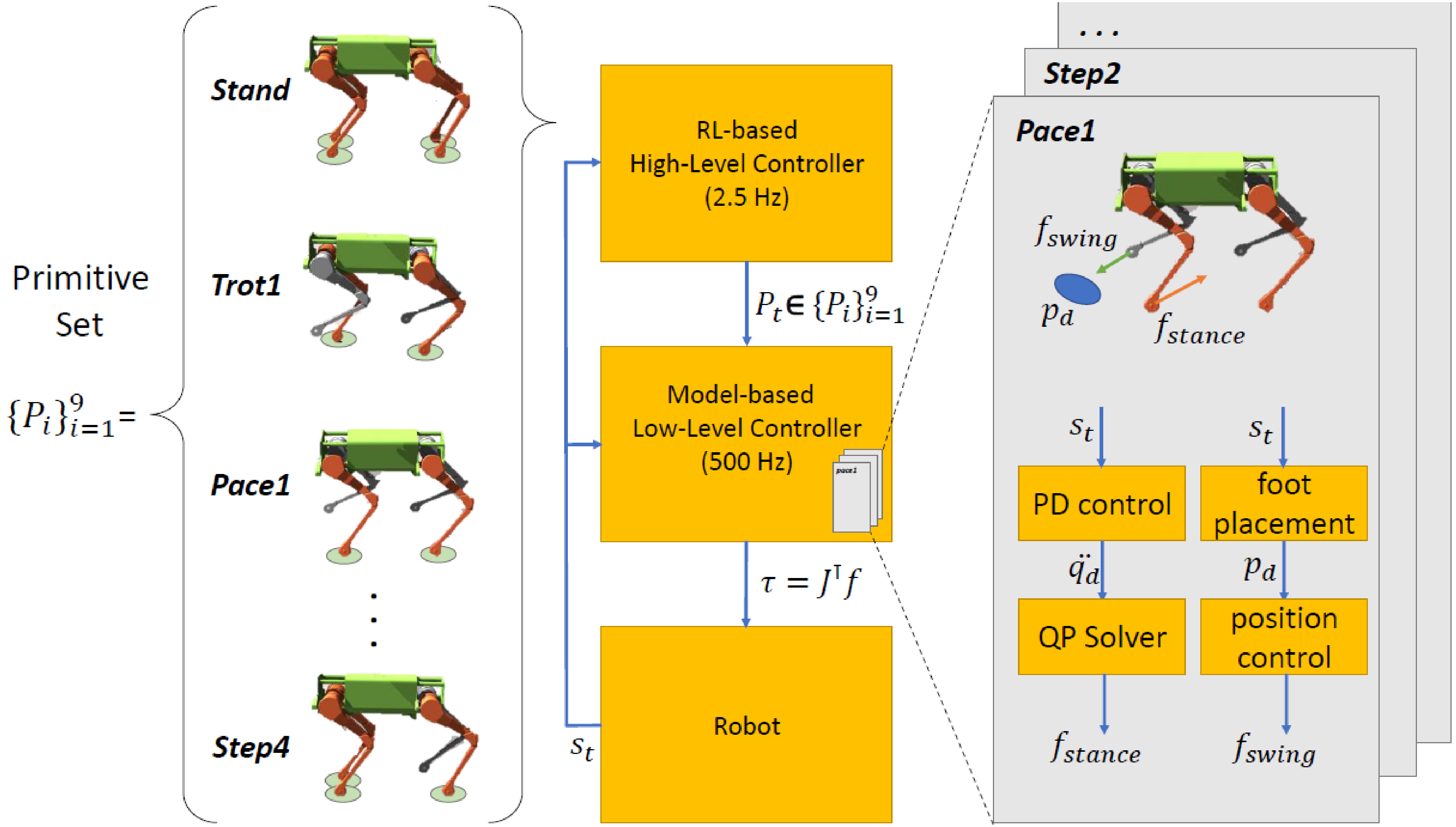




Control of Dynamic Gaits for a Quadrupedal Robot., Christian Gehring, Stelian Coros, Marco Hutter, Michael Bloesch, Markus A. Hoepflinger, Roland Siegwart, International Conference on Robotics and Automation(ICRA),2013
High-slope Terrain Locomotion for Torque-Controlled Quaruped Robots, Michele Focchi, Andrea del Prete, Ioannis Havoutis, Roy Featherstone, Darwin Caldwell, Claudio Semini, Autonomous Robots, 2017

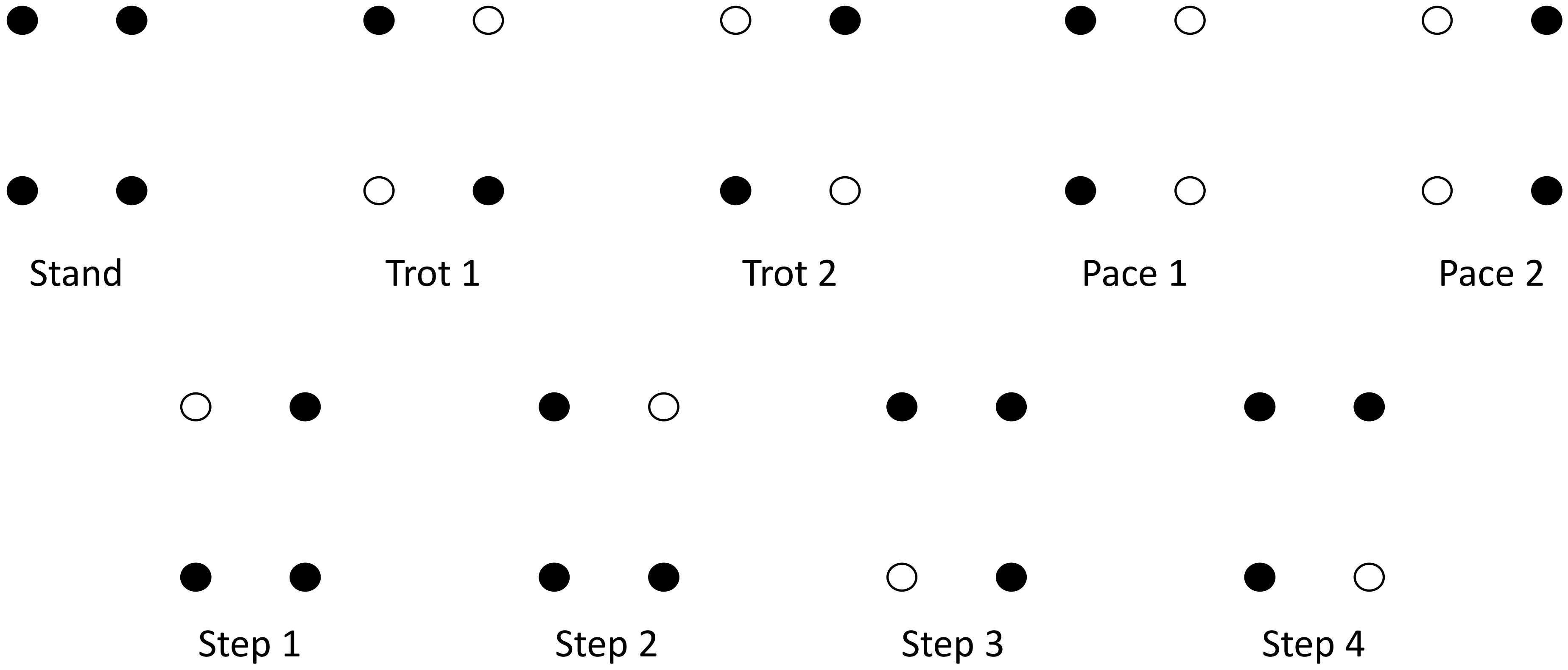
Paper Review

Controller overview



Paper Review

Primitive set



Paper Review

High level controller

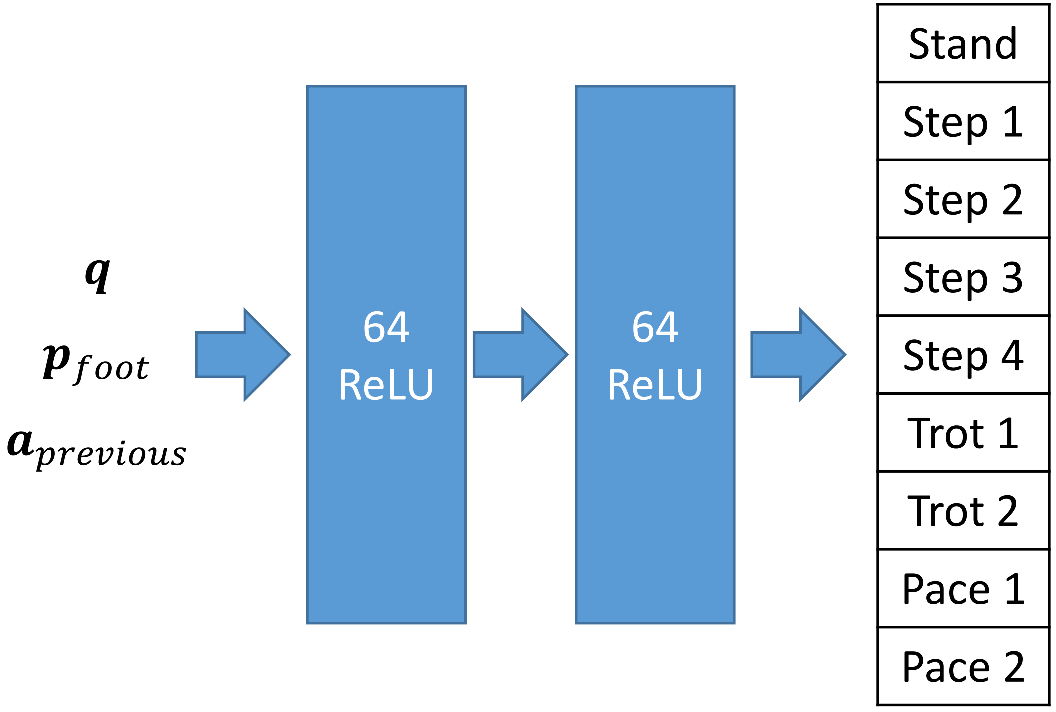
Simulator : Issac Gym

State Space
- Body pose $\boldsymbol{q}$ (exclude x, y linear position)
- Relative foot position $\boldsymbol{p}_{foot}$
- Previously-used primitive $\boldsymbol{a}_{previous}$

Action Space
- 9 Primitives

Reward

$$r = 1 - 0.0025\frac{1}{T}\Sigma\|\tau\|^2 - \frac{1}{T}\Sigma\left\|\dot{p}_{d,body} - \dot{p}_{body}\right\|^2$$



$$\boldsymbol{Q} = \{Q_1, Q_2, \cdots, Q_9\}$$

$$\mathrm{p_i} = \frac{\exp\left(-\nu\frac{Q_i}{Q_{max}}\right)}{\sum_{j=1}^{9}\exp\left(-\nu\frac{Q_j}{Q_{max}}\right)}$$

Paper Review

- High level controller

B Q-Learning Algorithm

We use DQN like algorithm to train our high-level policy. Details are shown in Algorithm 1.

Algorithm 1: Q Learning

---

initialization Q-function parameters $\theta_1.\theta_2$ for $Q_{\theta_1}, Q_{\theta_2}$, empty replay buffer D ;
set target network parameters $\theta_{targ,1}, \theta_{targ,2} \leftarrow \theta_1, \theta_2$ for $Q_{\theta_{targ,1}}, Q_{\theta_{targ,2}}$;
while *not done* **do**
  observe current state s ;
  sample action a based on Q-function;
  observe next state s', reward r and done signal d;
  store (s, a, r, d, s') in replay buffer D;
  if *d is True or time limit reached* **then**
    reset environment;
  end
  if *time to update* **then**
    for $j = 1, 2, \ldots$ *number of update* **do**
      sample batch of transition data $B = \{s, a, r, d, s'\}$;
      compute $a' = \arg\max_a Q_\theta(s', a)$;
      compute target $q_{targ} = r + (1 - d)\gamma \min_{i=1,2}(Q_{\theta_{targ,i}}(s', a'))$;
      update θ_1, θ_2 by taking gradient descent w.r.t the objective function
      $\frac{1}{|B|}\sum_{(s,a,r,d,s')\in B}((Q_{\theta_1}(s,a) - q_{targ})^2 + (Q_{\theta_2}(s,a) - q_{targ})^2)$;
      if $j \mod 2 = 1$ **then**
        $\theta_{targ,1} \leftarrow \rho\theta_{targ,1} + (1 - \rho)\theta_1$;
        $\theta_{targ,2} \leftarrow \rho\theta_{targ,2} + (1 - \rho)\theta_2$;
      end
    end
  end
end

Paper Review

Low level controller

Base Pose Control

$$\min_{\boldsymbol{f}} ||\mathbf{M}\boldsymbol{f} - \tilde{\boldsymbol{g}} - \ddot{\boldsymbol{q}}_{\boldsymbol{d}}||\boldsymbol{Q} + ||\boldsymbol{f}||\boldsymbol{R}$$

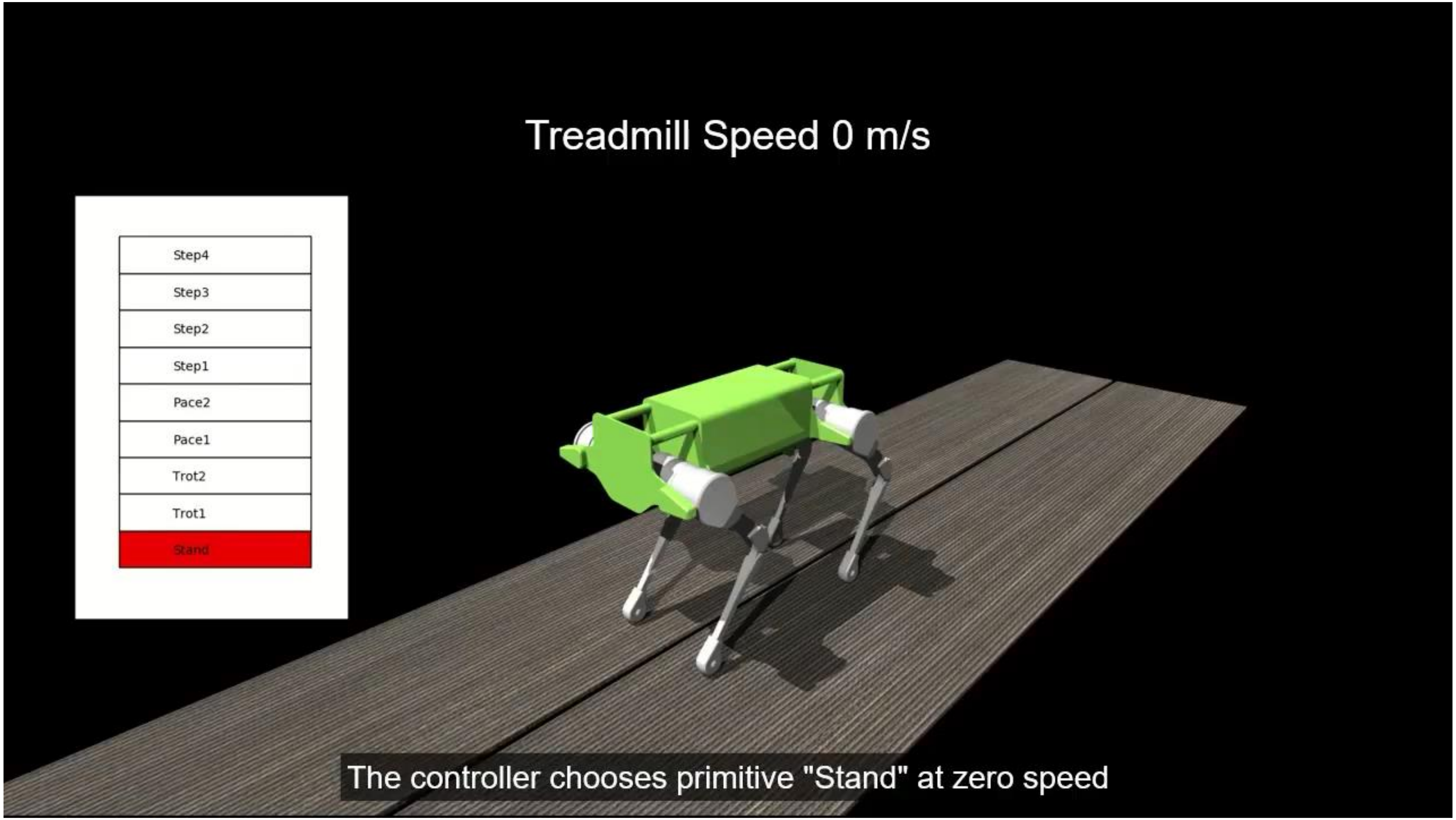
$$subject\ to\ f_{z,i} \geq f_{z,min}\ \ if\ P_{t,i}\ is\ Stance$$
$$f_{z,i} = 0\ if\ P_{t,i}\ is\ Swing$$
$$-\mu f_x \leq f_z \leq \mu f_x$$
$$-\mu f_y \leq f_z \leq \mu f_y$$

Swing Foot Control

$$p_{d,i} = p_{0,i} + k(\dot{p}_{body} - \dot{p}_{d,body})$$
$$f_i = k_{p,i}(p_{d,i} - p_i) - k_{d,i}\dot{p}_i$$

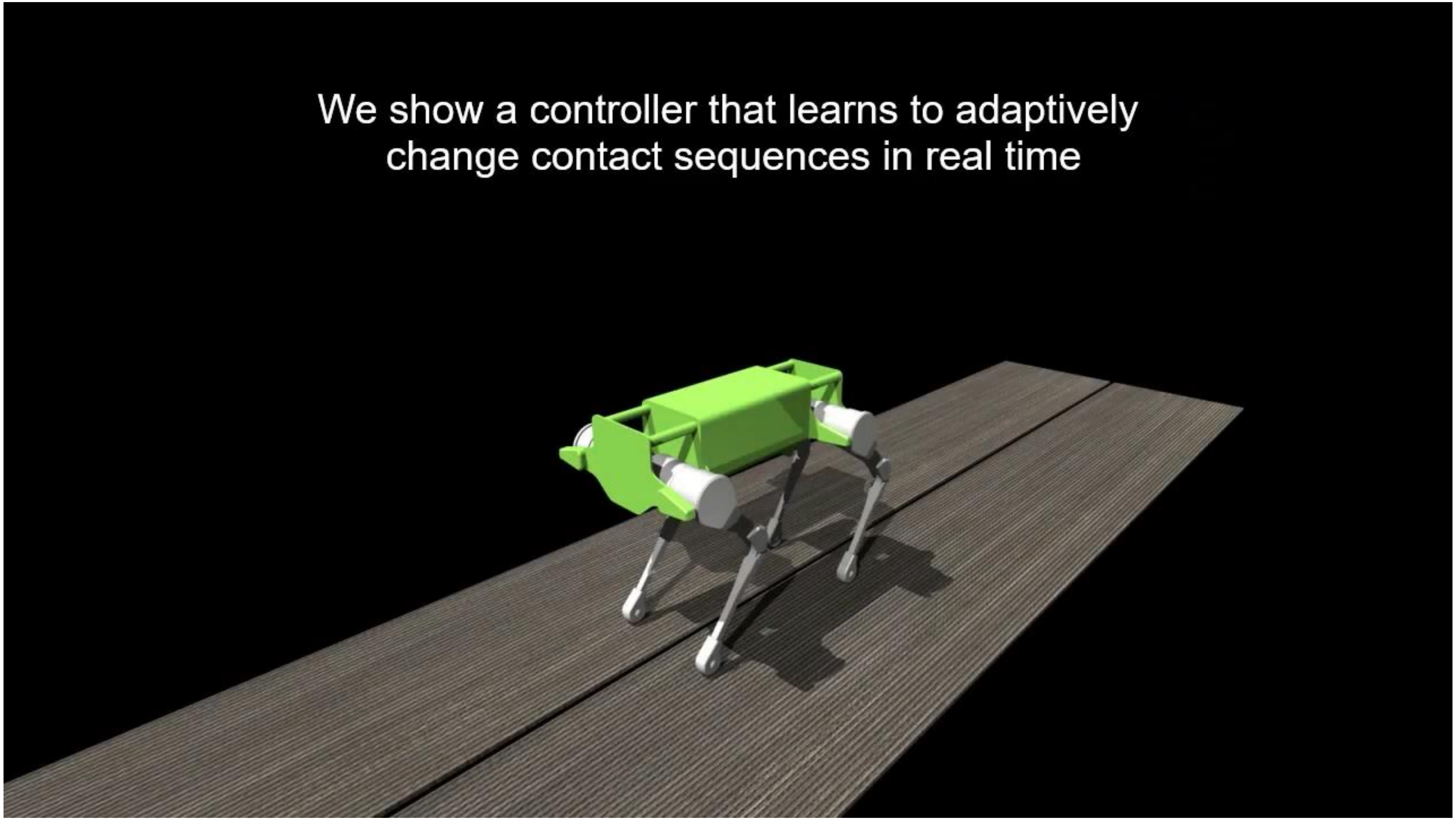
Paper Review

Result



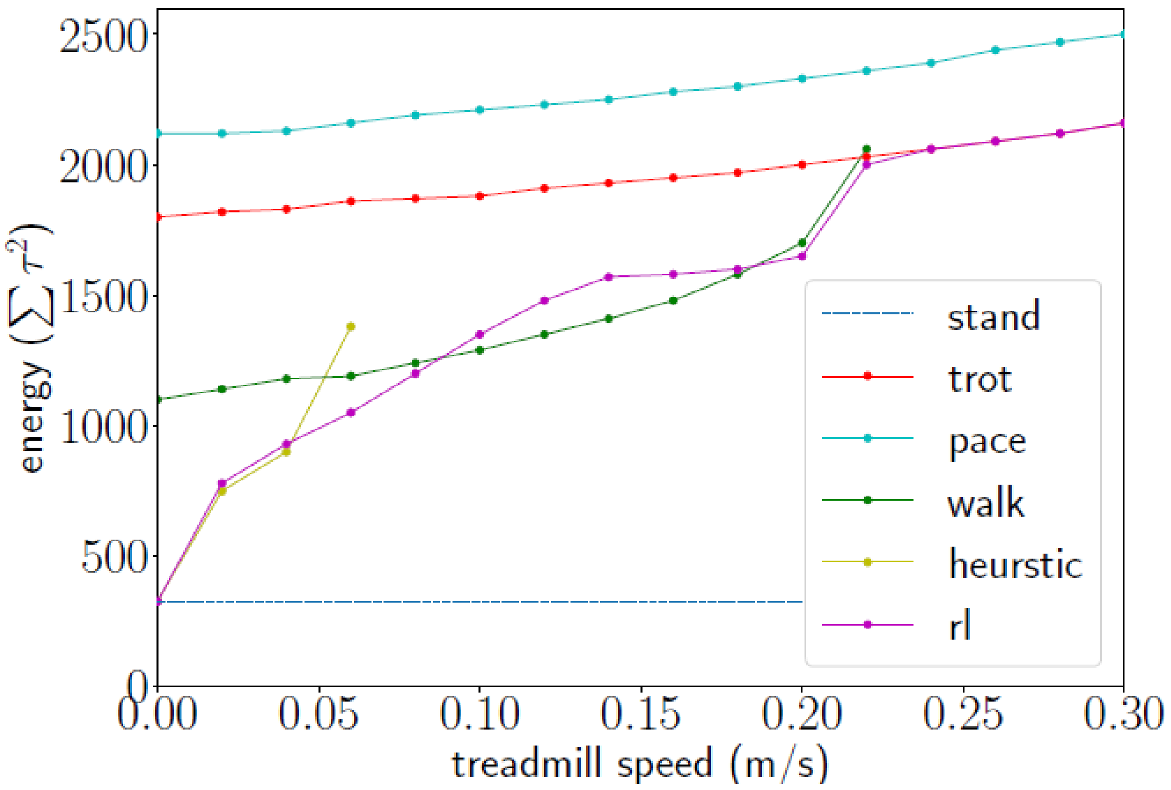
Paper Review

Result

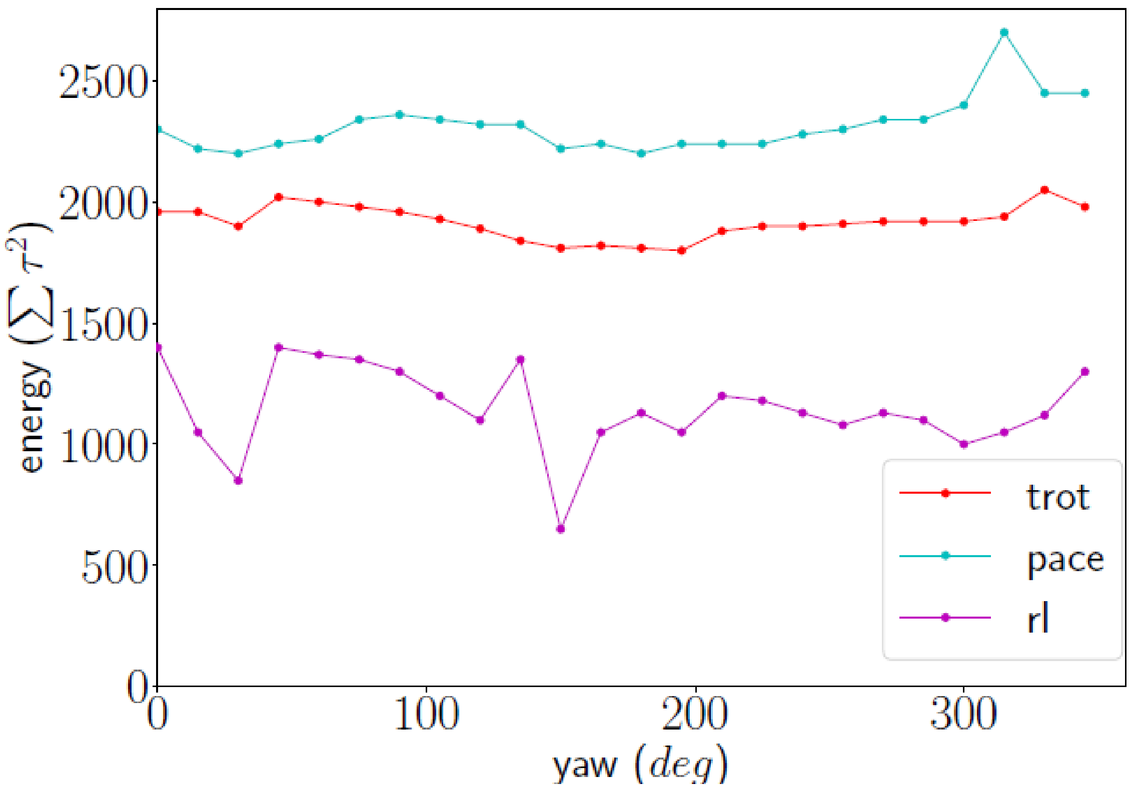


Paper Review

Result



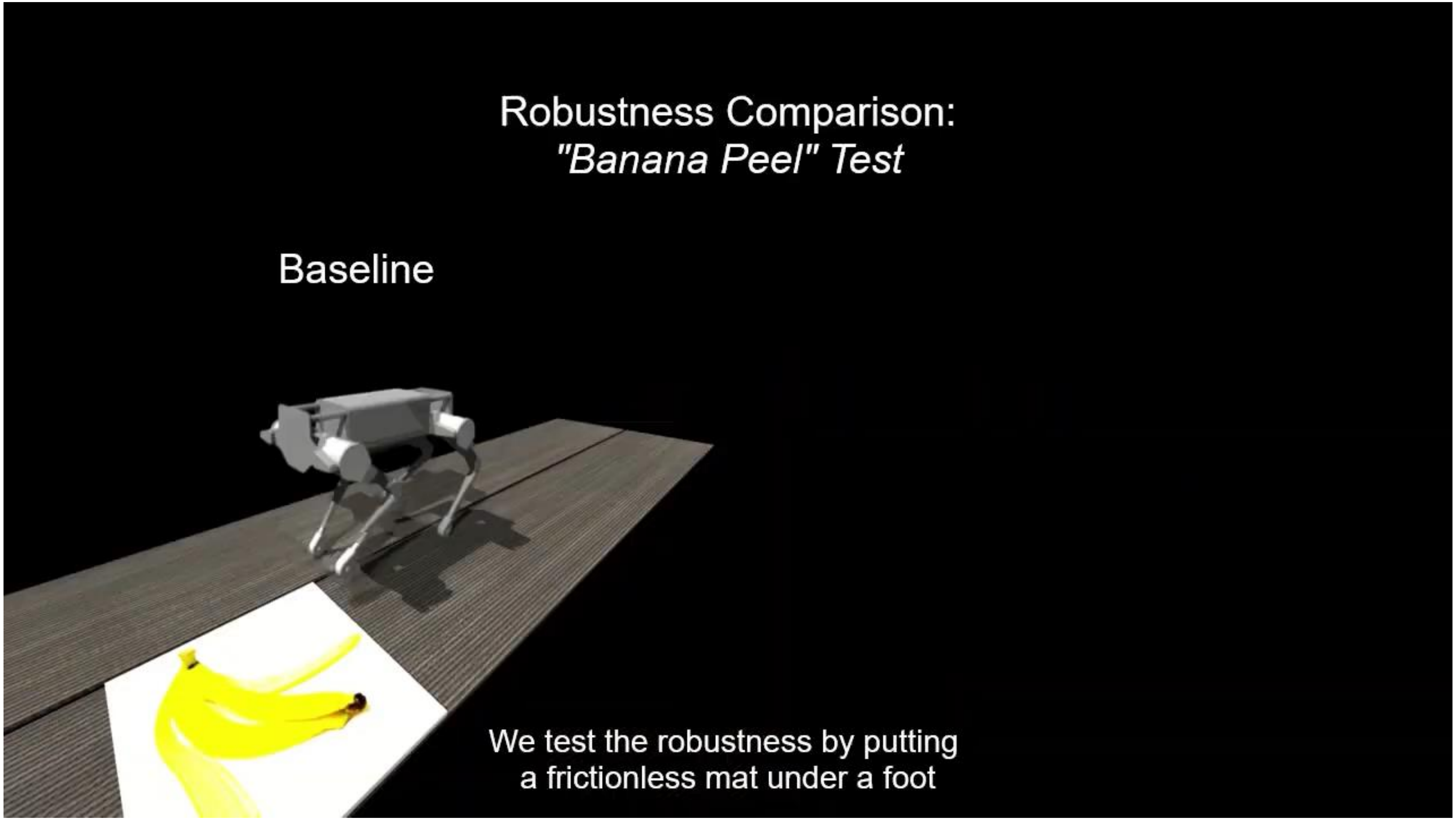
(a) Energy comparison over different speeds.



(b) Energy comparison over different yaws.

Paper Review

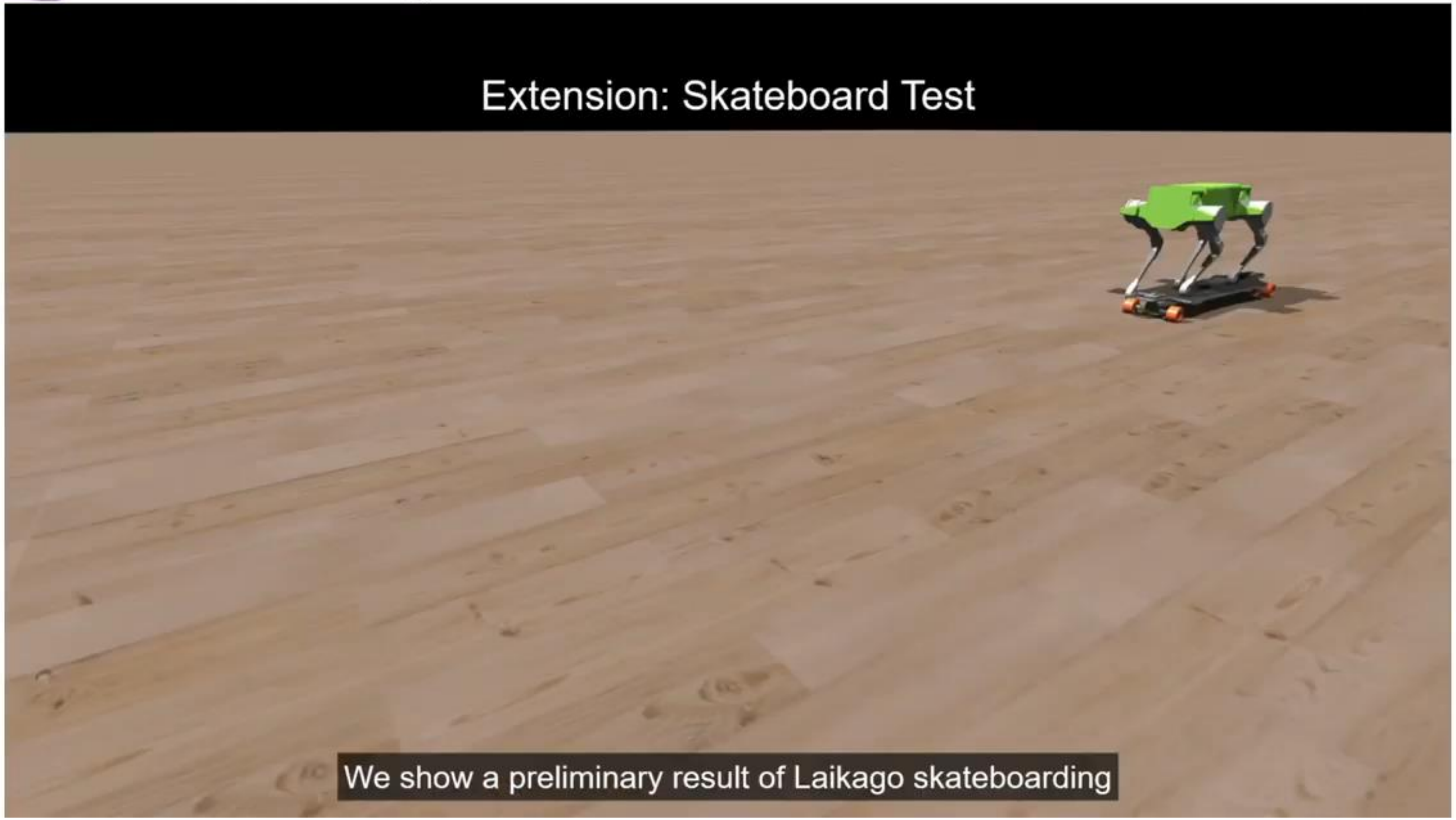
- Zero-shot adaptation



Paper Review

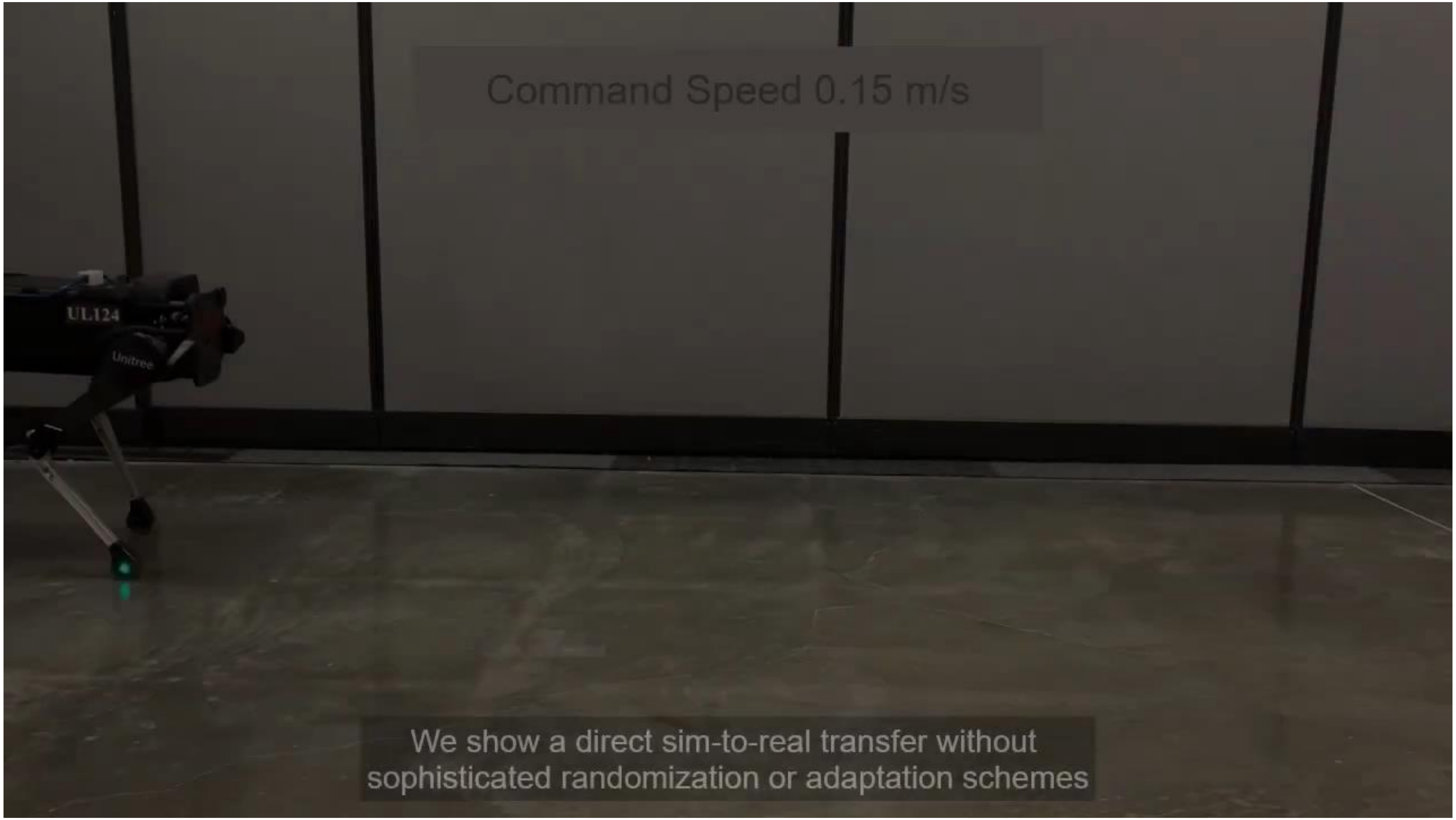
- Zero-shot adaptation

Conference on Robot Learning amazon robotics Aurora TOYOTA RESEARCH INSTITUTE WAYMO



Paper Review

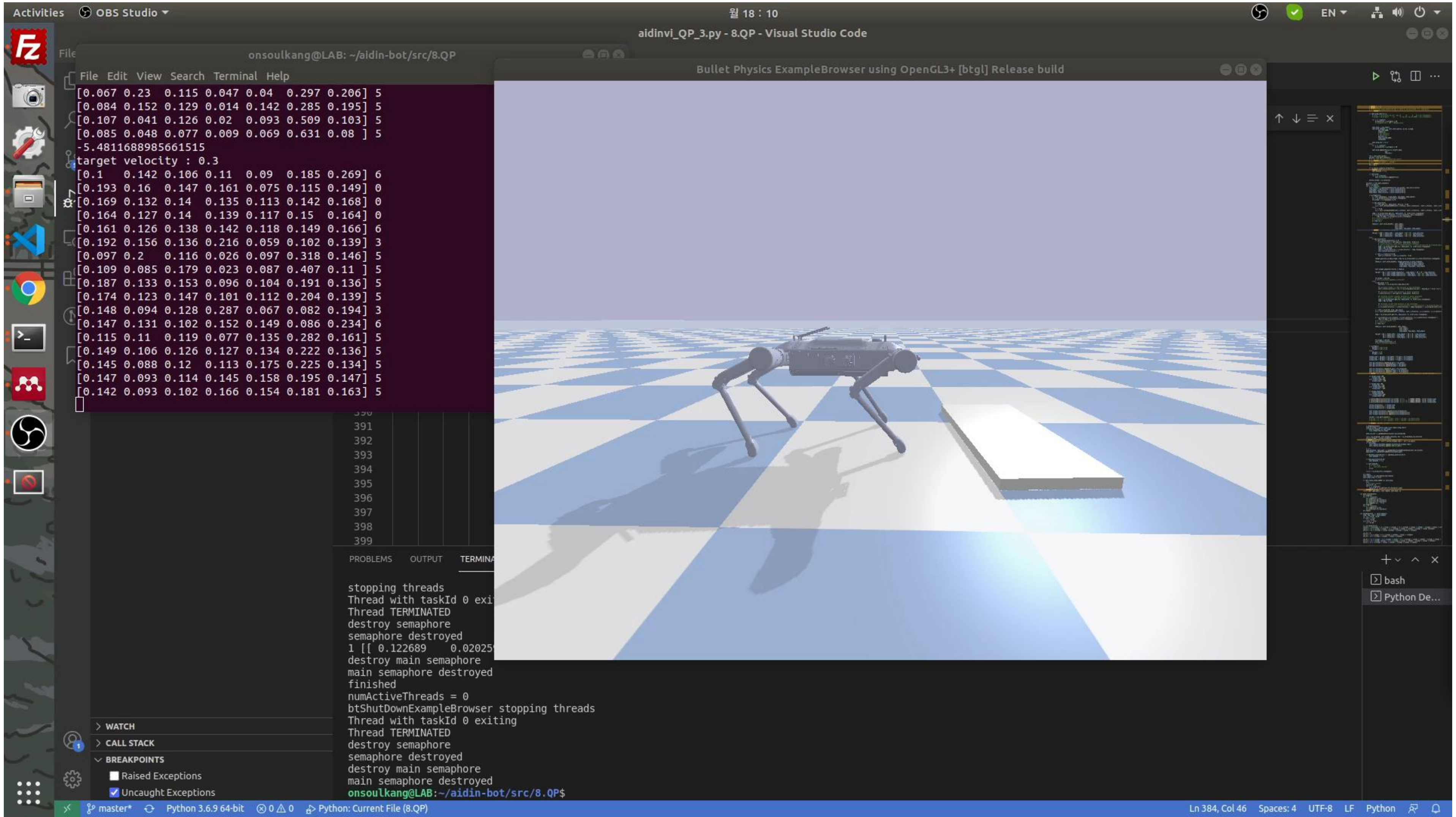
- Sim-to-real test



Implementation

Environment

PyBullet, PyTorch, Gym



Q & A

Thank you for your attention