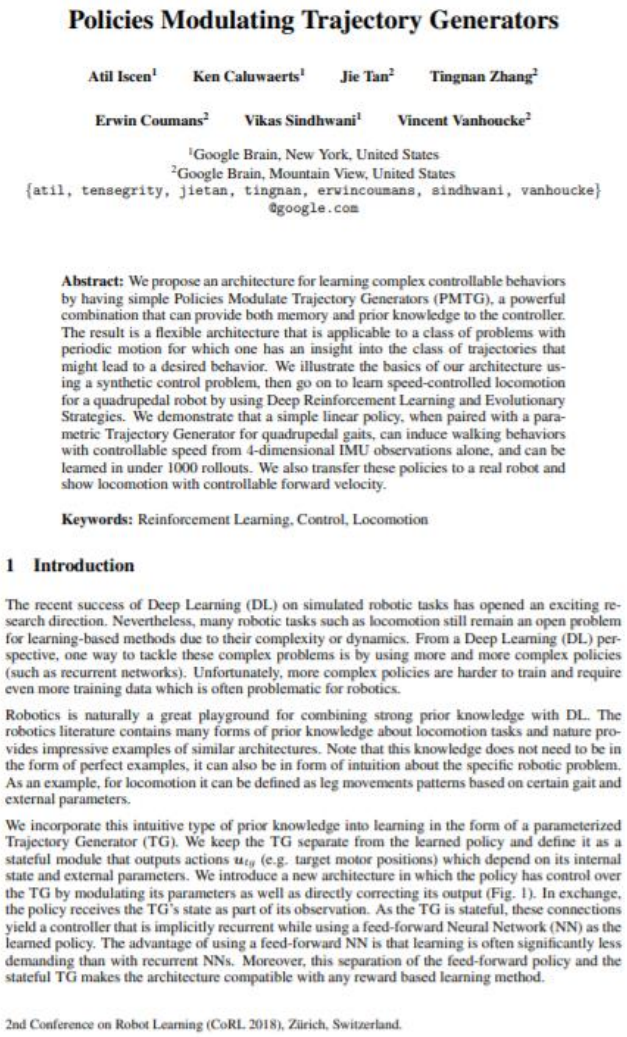
Policies Modulating Trajectory Generators

Paper Review

- Policies Modulating Trajectory Generators



Policies Modulating Trajectory Generators

Atil Iscen[1] Ken Caluwaerts[1] Jie Tan[2] Tingnan Zhang[2]

Erwin Coumans[2] Vikas Sindhwani[1] Vincent Vanhoucke[2]

[1]Google Brain, New York, United States
[2]Google Brain, Mountain View, United States
{atil, tensegrity, jietan, tingnan, erwincoumans, sindhwani, vanhoucke}
@google.com

Abstract: We propose an architecture for learning complex controllable behaviors by having simple Policies Modulate Trajectory Generators (PMTG), a powerful combination that can provide both memory and prior knowledge to the controller. The result is a flexible architecture that is applicable to a class of problems with periodic motion for which one has an insight into the class of trajectories that might lead to a desired behavior. We illustrate the basics of our architecture using a synthetic control problem, then go on to learn speed-controlled locomotion for a quadrupedal robot by using Deep Reinforcement Learning and Evolutionary Strategies. We demonstrate that a simple linear policy, when paired with a parametric Trajectory Generator for quadrupedal gaits, can induce walking behaviors with controllable speed from 4-dimensional IMU observations alone, and can be learned in under 1000 rollouts. We also transfer these policies to a real robot and show locomotion with controllable forward velocity.

Keywords: Reinforcement Learning, Control, Locomotion

1 Introduction

The recent success of Deep Learning (DL) on simulated robotic tasks has opened an exciting research direction. Nevertheless, many robotic tasks such as locomotion still remain an open problem for learning-based methods due to their complexity or dynamics. From a Deep Learning (DL) perspective, one way to tackle these complex problems is by using more and more complex policies (such as recurrent networks). Unfortunately, more complex policies are harder to train and require even more training data which is often problematic for robotics.

Robotics is naturally a great playground for combining strong prior knowledge with DL. The robotics literature contains many forms of prior knowledge about locomotion tasks and nature provides impressive examples of similar architectures. Note that this knowledge does not need to be in the form of perfect examples, it can also be in form of intuition about the specific robotic problem. As an example, for locomotion it can be defined as leg movements patterns based on certain gait and external parameters.

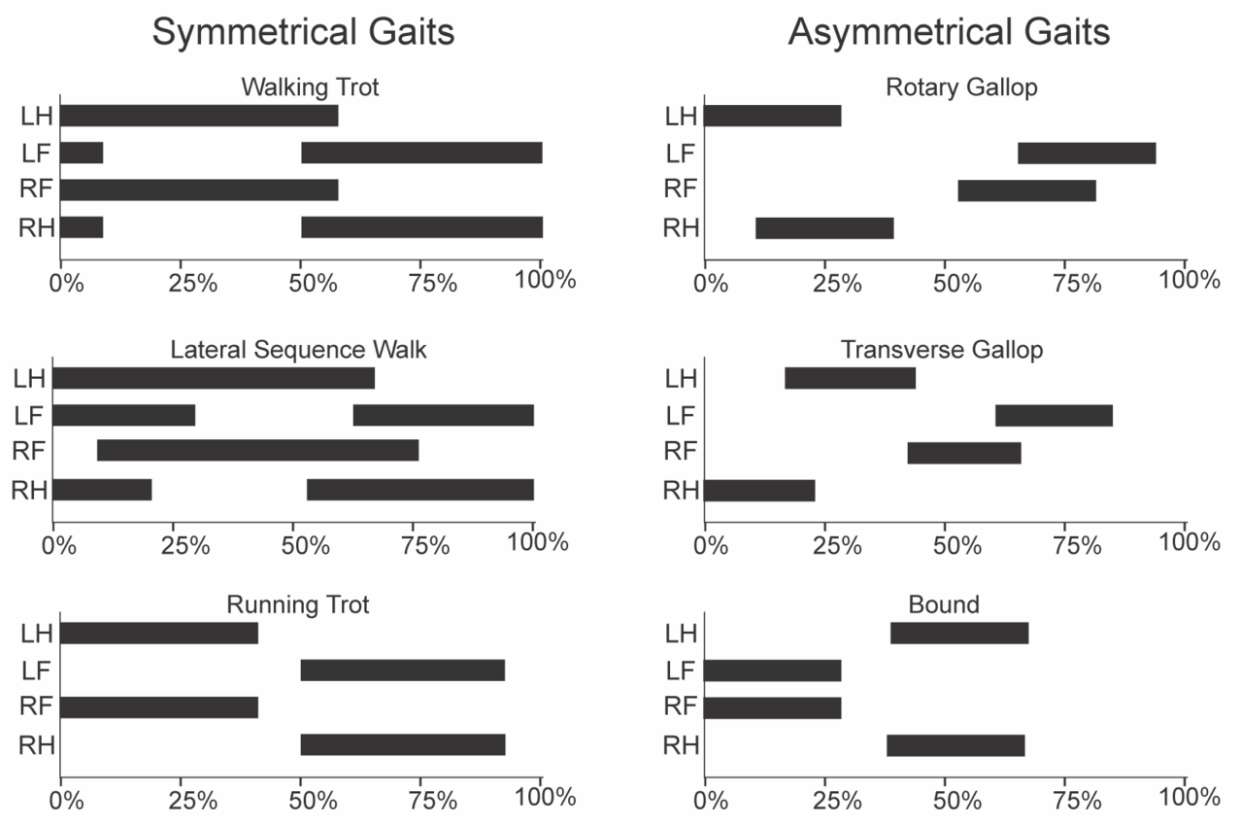
We incorporate this intuitive type of prior knowledge into learning in the form of a parameterized Trajectory Generator (TG). We keep the TG separate from the learned policy and define it as a stateful module that outputs actions u_{tg} (e.g. target motor positions) which depend on its internal state and external parameters. We introduce a new architecture in which the policy has control over the TG by modulating its parameters as well as directly correcting its output (Fig. 1). In exchange, the policy receives the TG's state as part of its observation. As the TG is stateful, these connections yield a controller that is implicitly recurrent while using a feed-forward Neural Network (NN) as the learned policy. The advantage of using a feed-forward NN is that learning is often significantly less demanding than with recurrent NNs. Moreover, this separation of the feed-forward policy and the stateful TG makes the architecture compatible with any reward based learning method.

2nd Conference on Robot Learning (CoRL 2018), Zürich, Switzerland.

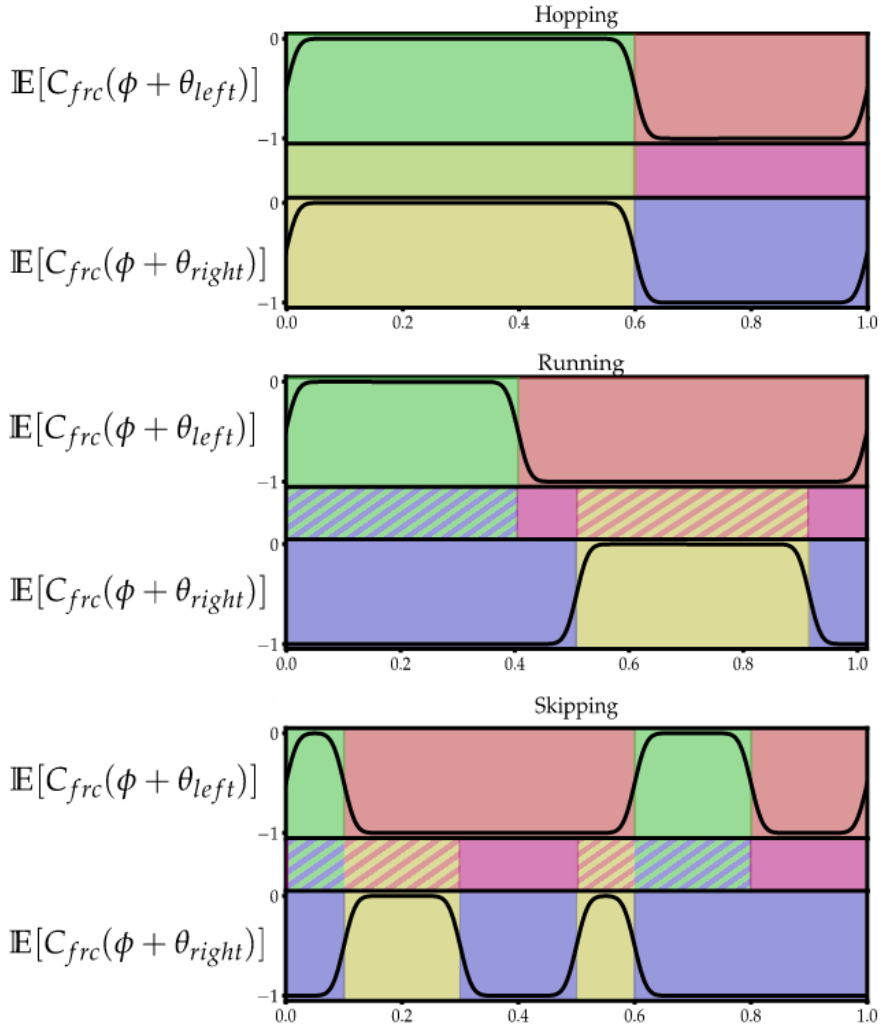
- Conference on Robot Learning(CoRL), 2018
- Atil Iscen[1], Ken Caluwaerts[1], Jie Tan[1], Tingnan Zhang[2], Erwin Coumans[2], Vikas Sindhwani[2], Vincent Vanhouke[2]
- [1]Google Brain, New York
  [2]Google Brain, Mountain View
- Propose an architecture for learning complex controllable behaviors by having simple Policies Modulate Trajectory Generators (PMTG), a powerful combination that can provide both memory and prior knowledge to the controller.

Introduction

- Locomotion is periodic and structured motion



Quadruped Gaits



Biped Gaits

https://www.wikiwand.com/en/Gait

Sim-to-Real Learning of All Common Bipedal Gaits via Periodic Reward Composition, Jonah Siekmann et. al., IEEE International Conference on Robotics and Automation (ICRA), 2021.

Introduction

- Reference trajectories from previous researches

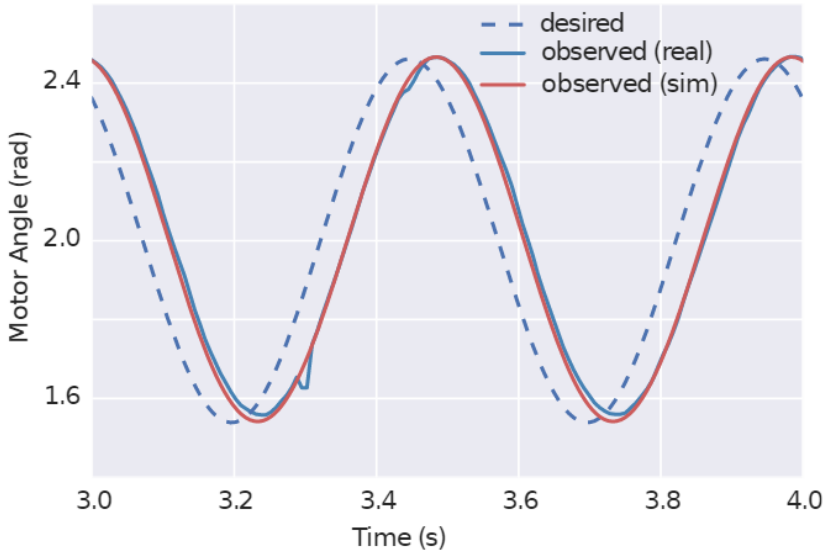
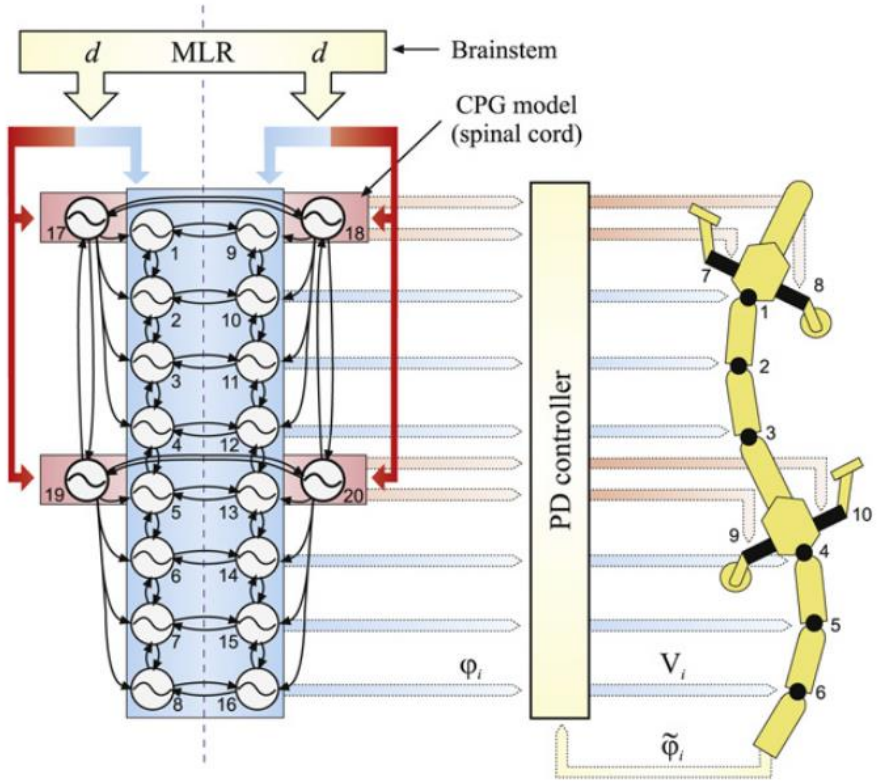


Fig. 4: Comparison of the simulated motor trajectory (red) with the ground truth (blue).

Simple sine wave

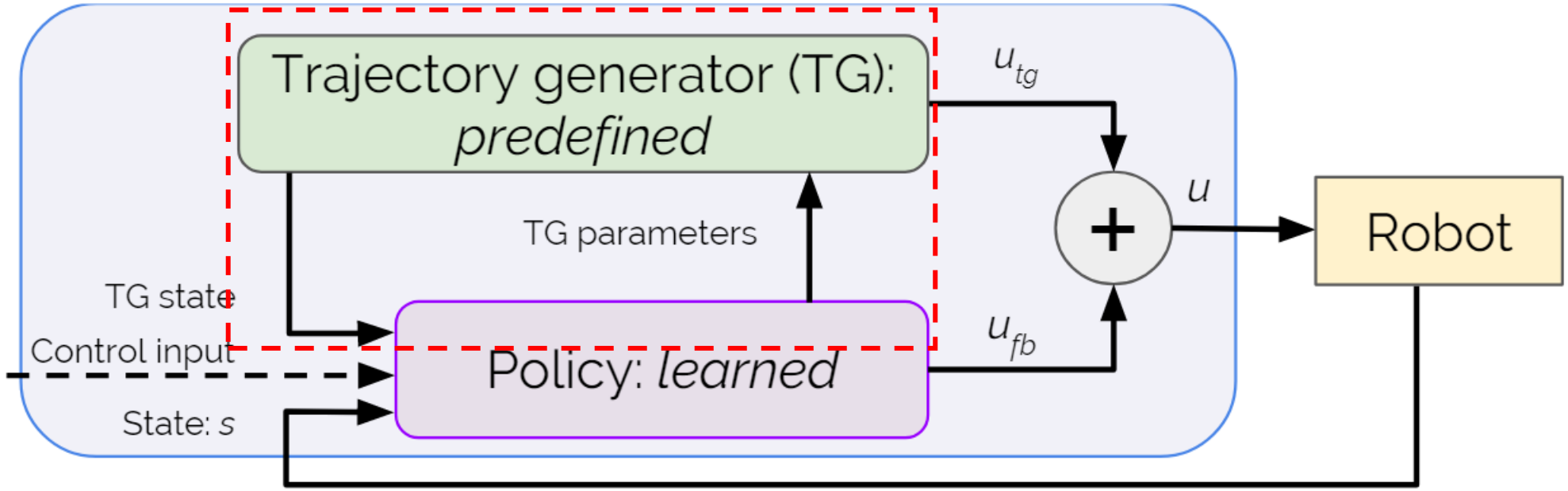


Central Pattern Generator(CPG)

Sim-to-Real: Learning Agile Locomotion For Quadruped Robots, Jie Tan et. al., Robotics: Science and Systems (RSS). 2018.
Central pattern generators for locomotion control in animals and robots: A review, A.J.Ijspeert, Neural Networks, 2008.

Paper Review

- Overview of PMTG



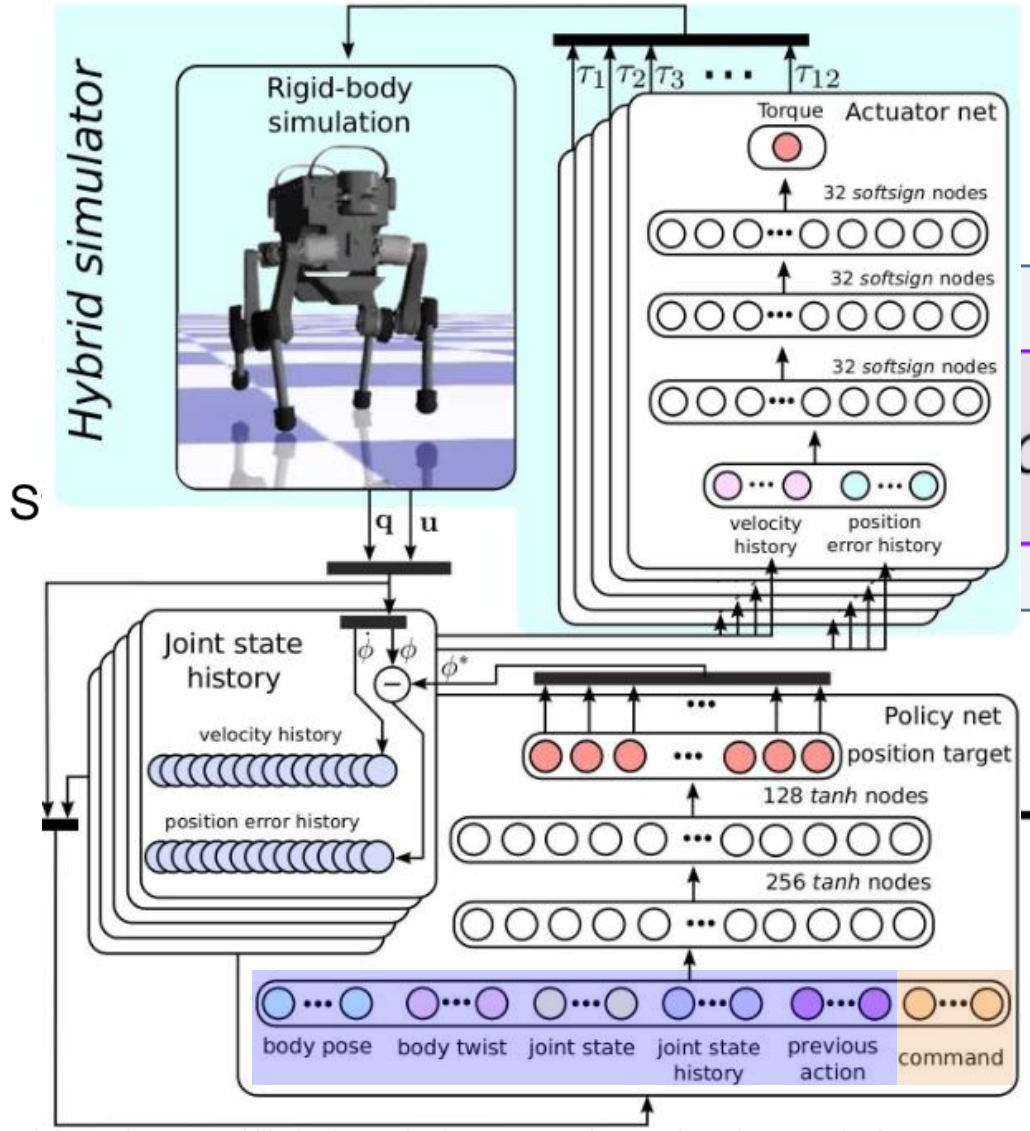
Paper Review

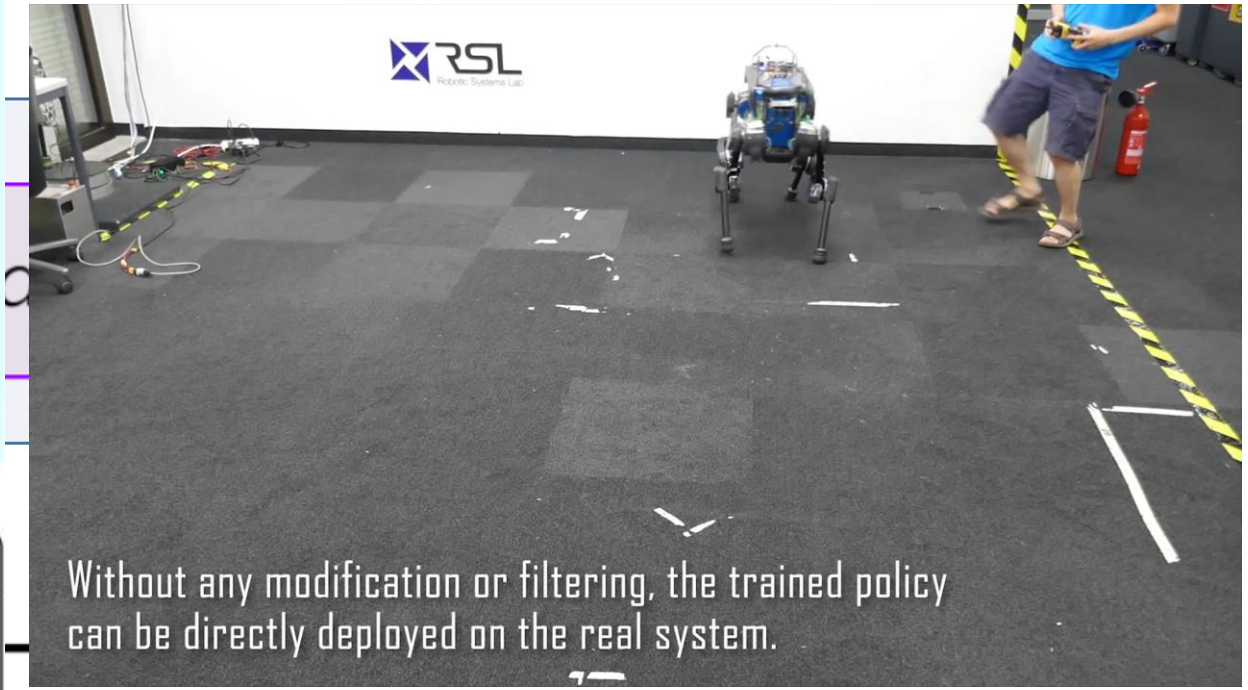
- Previous researches method

Sim-to-Real: Learning Agile Locomotion For Quadruped Robots, Jie Tan et. al., Robotics: Science and Systems (RSS). 2018.

Paper Review

- Previous researches method

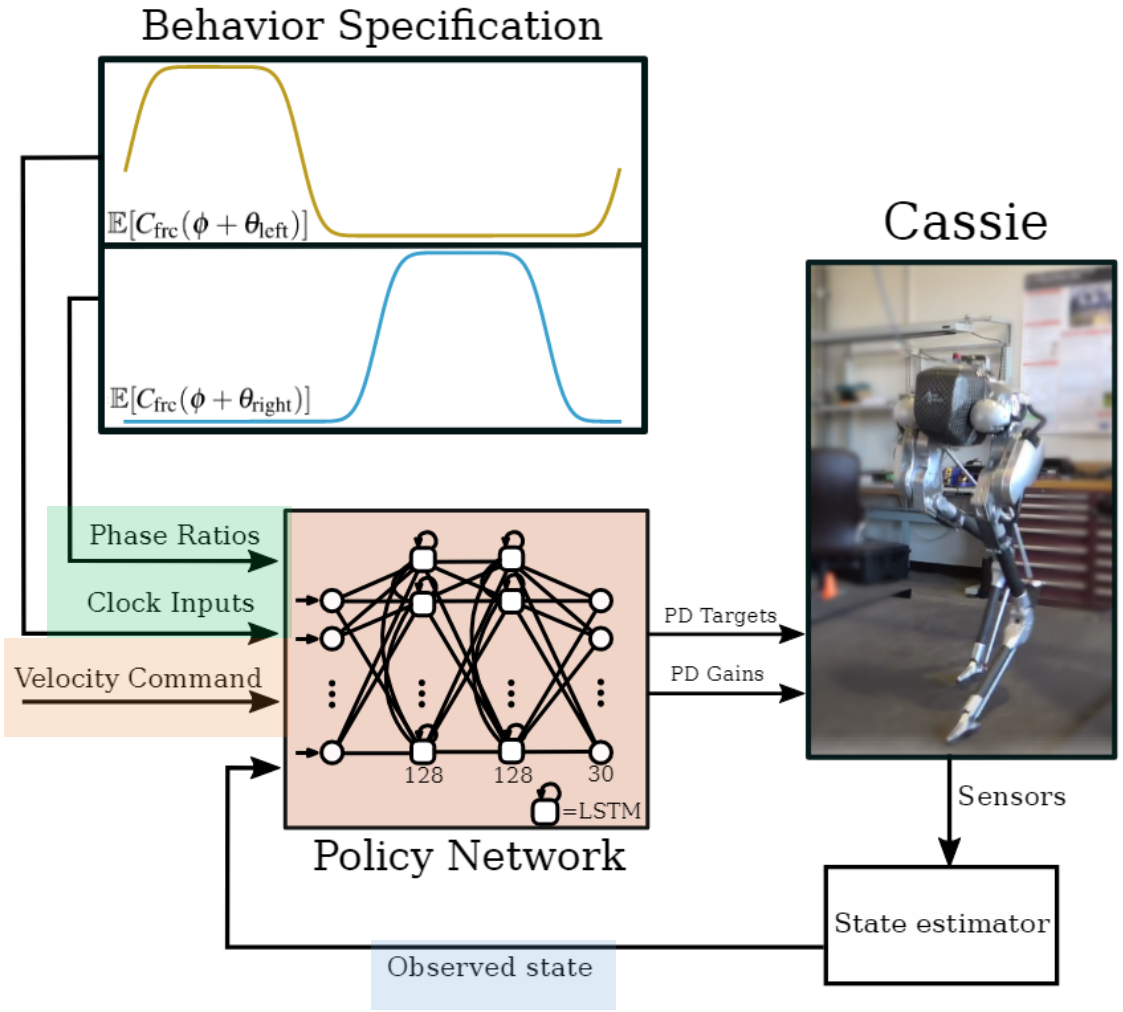




Learning agile and dynamic motor skills for legged robots, Hwangbo et. al., Science Robotics, 2019

Paper Review

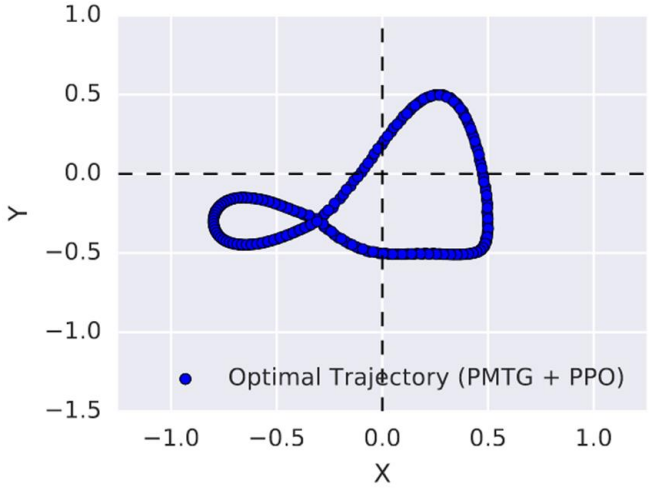
- Previous researches method



Sim-to-Real Learning of All Common Bipedal Gaits via Periodic Reward Composition, Jonah et. al., IEEE International Conference on Robotics and Automation(ICRA), 2021

Paper Review

- Synthetic control problem



(a) Optimal (learned) behavior.

Case 1. Vanila PPO

- Input : Current position(x,y)
- Output : Next position(x,y)

Case 2. Vanila PPO + Time signal

- Input : Current position(x,y), Time signal
- Output : Next position(x,y)

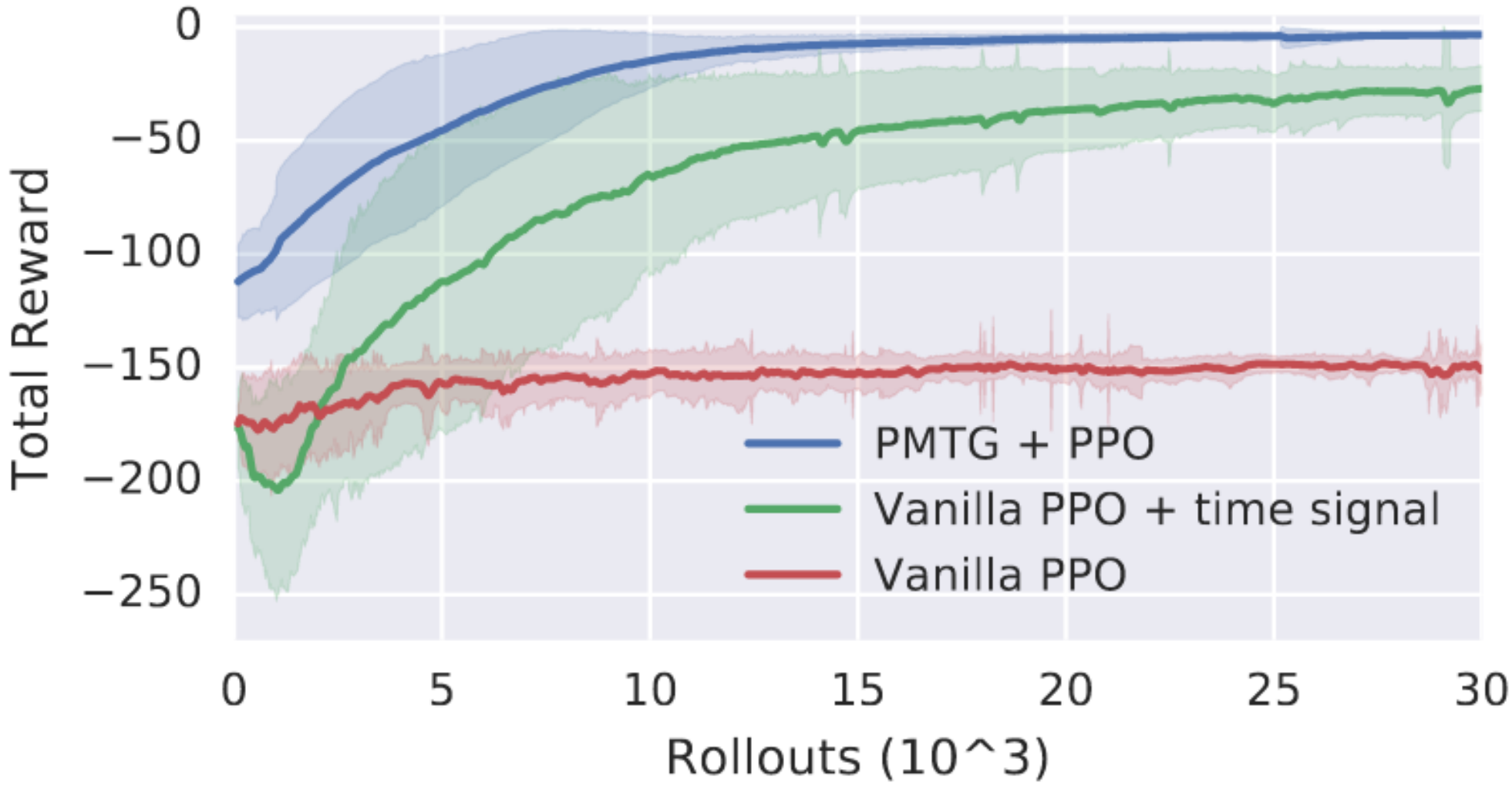
Case 3. PMTG + PPO

- Input : Current position(x,y), TG phase
- Output : Next position(x,y), TG Parameter
- TG : $u_{tg}(a_x, a_y) = \begin{bmatrix} a_x \sin(2\pi t) \\ \frac{a_y}{2} \sin(2\pi t) \cos(2\pi t) \end{bmatrix}$

Reward : $-\|p_d - p_{net}\|$

Paper Review

- Synthetic control problem



Paper Review

- Quadruped locomotion

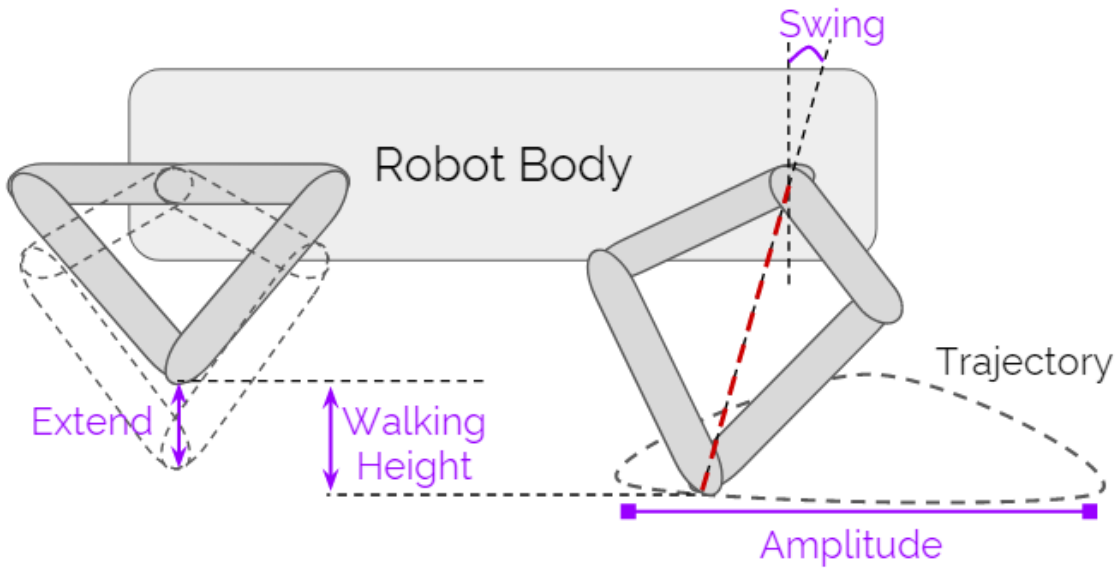


Figure 4: Illustration of robot leg trajectories generated by the TG.

$$u_{tg} = \begin{bmatrix} S(t) \\ E(t) \end{bmatrix} = \begin{bmatrix} C_s + \alpha_{tg}\cos(t') \\ h_{tg} + A_e \sin(t') + \theta\cos(t') \end{bmatrix}$$

$$t' = \begin{cases} \dfrac{\phi_{leg}}{2(1-\beta)} & if\ 0 < \phi_{leg} < 2\pi\beta \\ 2\pi - \dfrac{2\pi - \phi_{leg}}{2\beta} & otherwise \end{cases}$$

$$\phi_t = \phi_{t-1} + 2\pi f_{tg}\Delta t \ mod\ 2\pi$$

C_s : $Center\ of\ swing\ DOF$
h_{tg} : $Center\ of\ extension\ DOF$
α_{tg} : $Amplitude\ of\ swing\ signal(Stride)$
A_e : $Amplitude\ of\ extension\ signal(ground\ clearance)$
θ : $Extension\ difference\ between$
$the\ end\ of\ swing\ and\ the\ end\ of\ stance$

Paper Review

- Quadruped locomotion

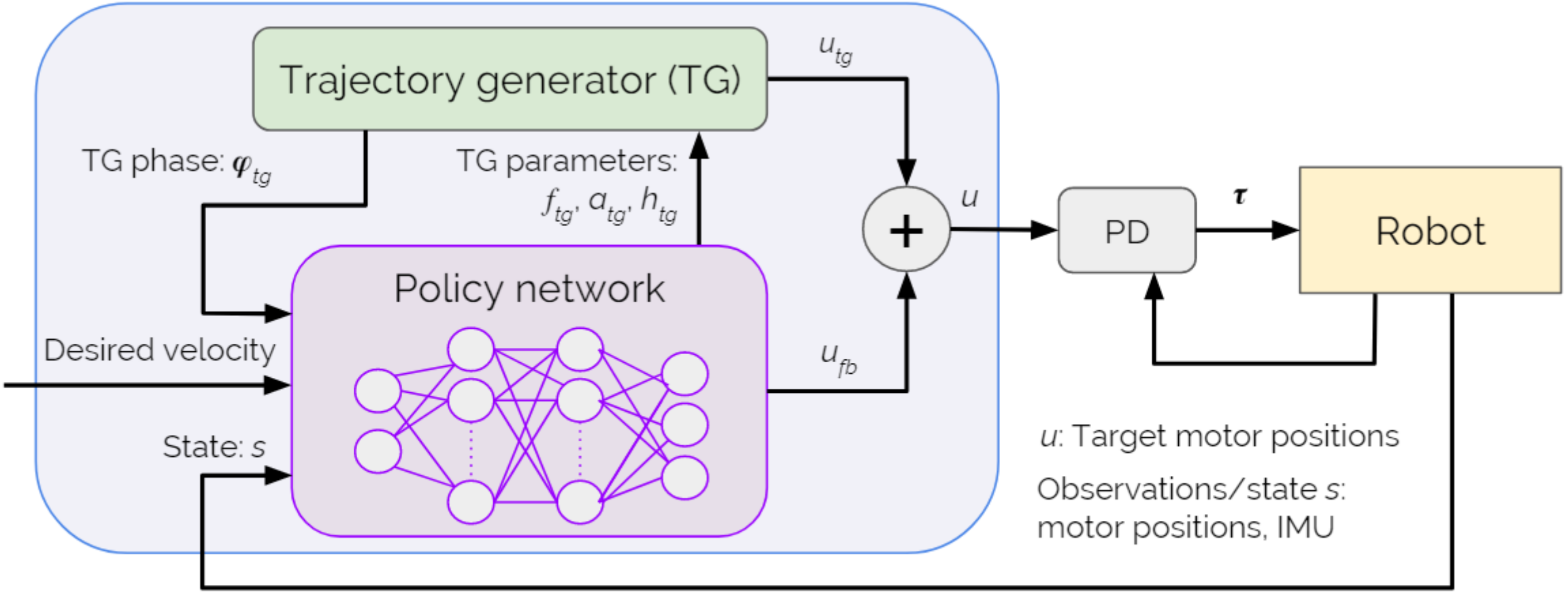


Figure 5: Adaptation of PMTG to the quadruped locomotion problem.

Paper Review

- Quadruped locomotion

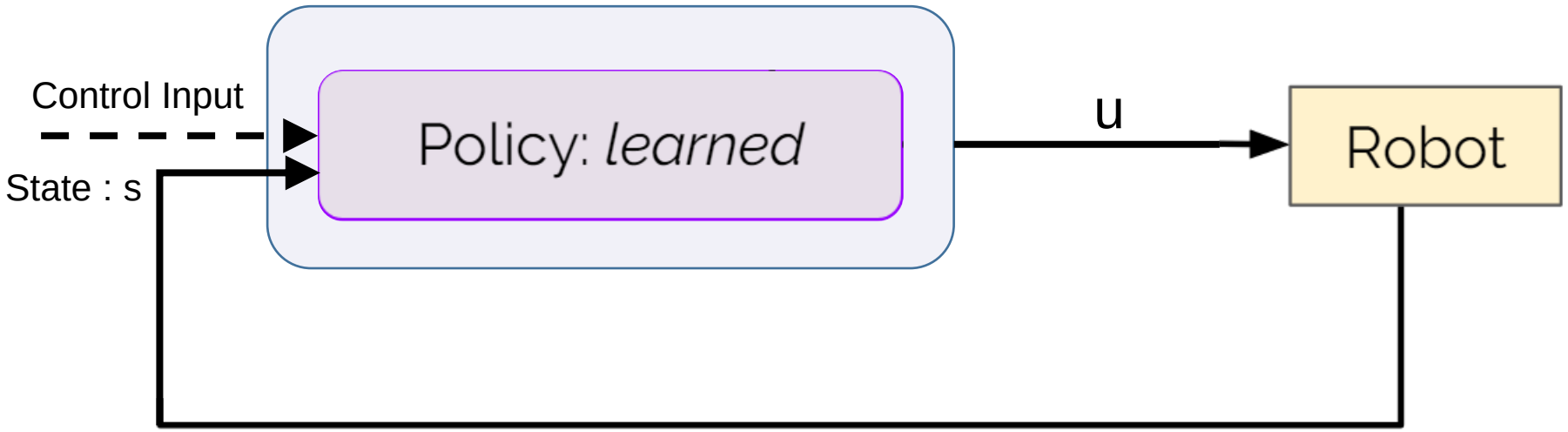
Paper Review

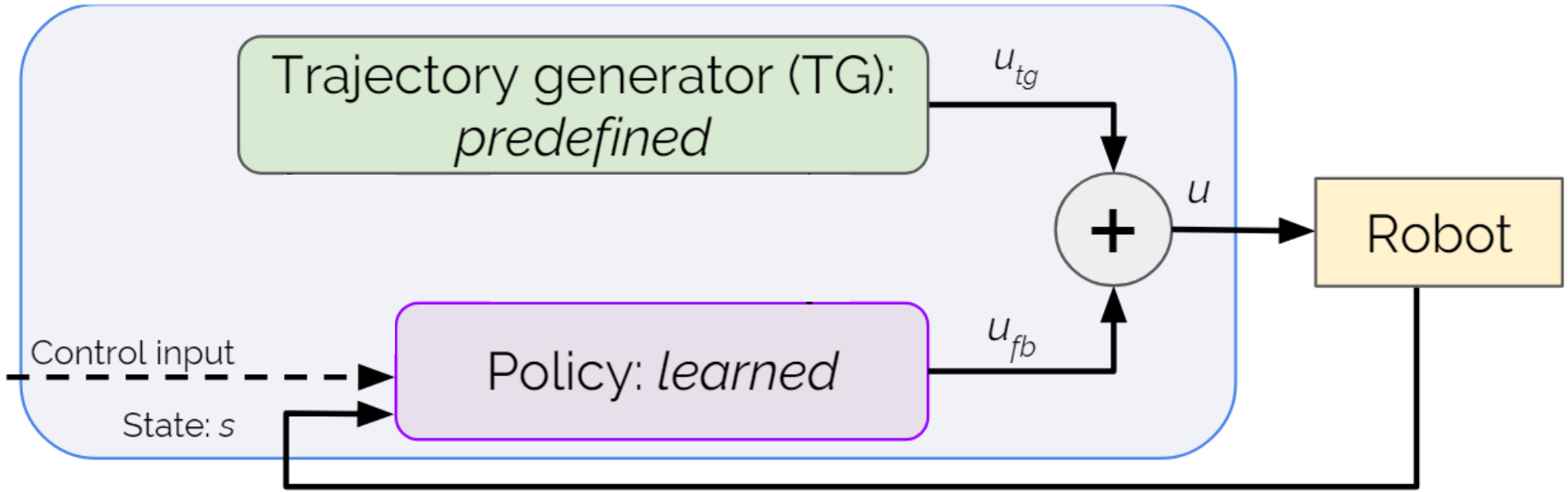
- The significance of PMTG

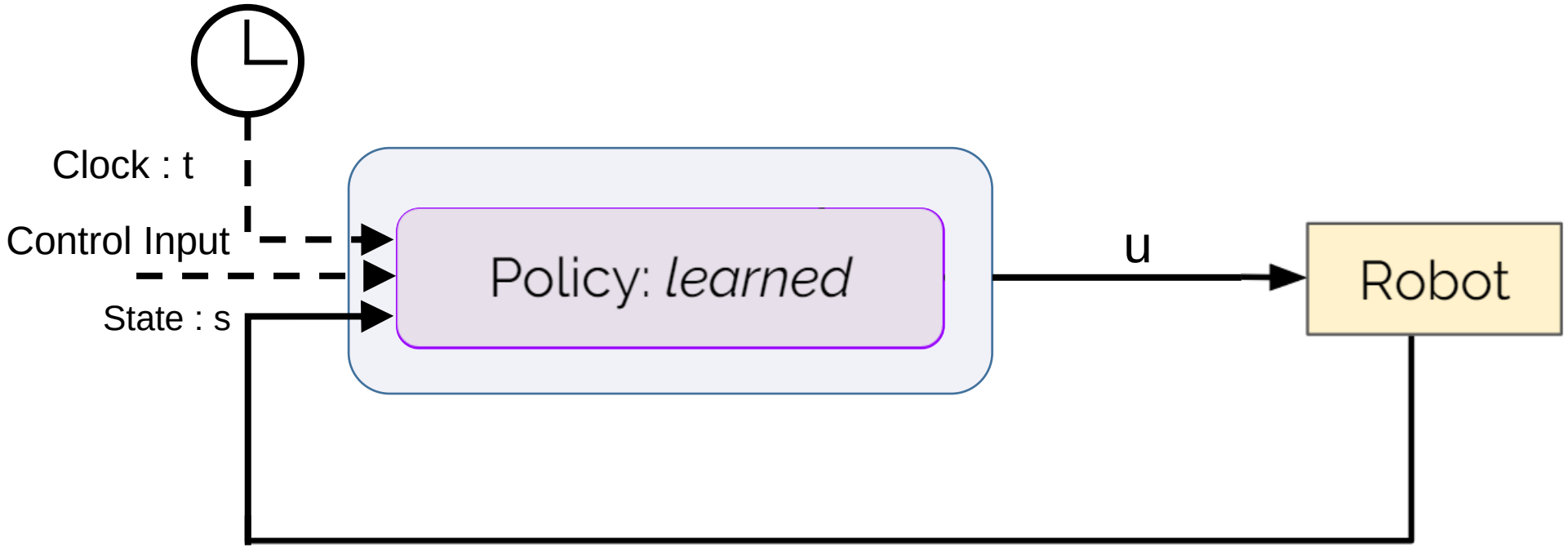
RMA: Rapid Motor Adaptation for Legged Robots, Kumar et. al., arXiv, 2021
Learning quadrupedal locomotion over challenging terrain, Lee et. al., Science Robotics, 2020

Q & A

Thank you for your attention


Control Input
State : s
Policy: learned
u
Robot


Trajectory generator (TG):
predefined
u_{tg}
u
+
Robot
Control input
Policy: learned
u_{fb}
State: s


Clock : t
Control Input
State : s
Policy: learned
u
Robot