Iscen et al. - 2018 - Policies Modulating Trajectory Generators

Fournier Cédric
Opreni Carlo
Leroy Hugo

November 2024

École polytechnique fédérale de Lausanne

OUTLINE

Introduction

Technical Highlights

Training

Results

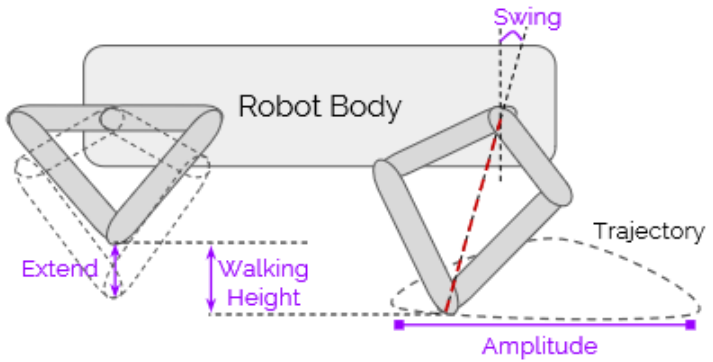
Pro and Cons

Experiments & Possible questions

Introduction

Short Summary:

- **Architecture**: Introduces **Policies Modulating Trajectory Generators (PMTG)**, combining simple policy with a **Trajectory Generator (TG).**



Executive Summary:

•**Robot Type**: Quadruped robot

•**Control Method**: A combined control approach where a learned policy modulates a predefined **TG**. The **learned policy** can influence parameters of the **TG**.

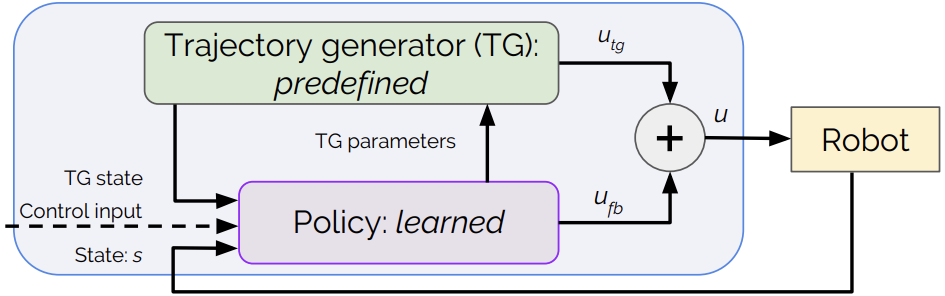
•**Design Method**: The **TG** design is based on a hand-tuned approach , and the policy is trained via a Proximal policy optimization (PPO)

•**Gait Types**: Walking, running, and bounding gaits were tested.

•**Sensors Used**: 4-dimensional IMU reading, which includes angular velocity and robot orientation

Technical Highlights

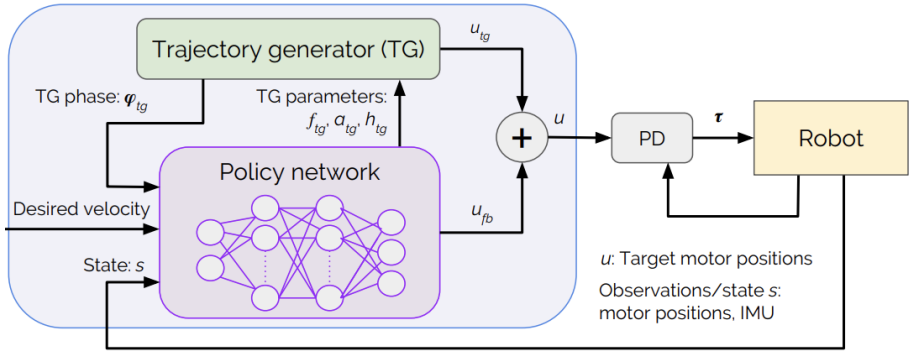
PMTG General Architecture



- **PMTG** architecture focuses on the interaction between a **learned policy** and a **predefined TG**
- The architecture incorporates prior knowledge through a parameterized **TG** that operates separately from the **learned policy**.
- The **TG**, as a stateful module, generates motor actions based on internal states and adjustable parameters.
- The **policy** modulates these parameters and can directly correct **TG** outputs, while observing the **TG**'s state
- This setup uses a feed-forward Neural Network (NN) for the **policy**

Technical Highlights

Adaptation on Quadruped Locomotion



- **Inputs**: The **policy** receives observations (s), desired velocity, and **TG** phase (phi*tg*) at each timestep.
- **TG Parameters**: Computes frequency (f*tg*), amplitude (a*tg*), and walking height (h*tg*) for the **TG**.
- **Leg Actions**: Calculates eight actions for leg positions (u*fb*), added to TG's calculated leg positions (u*tg*).
- **Control Tracking**: Motor positions are tracked by Proportional-Derivative controllers.
- **Time Modulation**: Policy-controlled frequency allows flexible, time-independent modulation of the **TG** phase.

Technical Highlights

The phase of the trajectory generator (between 0 and 2π) is defined as:

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

where f_{tg} defines the frequency of the trajectory generator. In PMTG architecture, f_{tg} is selected by the policy at each time step as an action.

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

- $S(t)$, and $E(t)$ are respectively the swing and extension of the leg as shown in Fig. 4.
- C_s defines the center for the swing DOF and extension DOF (in radians).
- h_{tg} defines the center for the extension DOF. Extension is represented in terms of rotation of the two motors in the opposite direction, hence the unit is also radians. Since all legs share the same h_{tg}, it corresponds to the walking height of the robot.
- α_{tg} defines the amplitude of the swing signal (in radians). This corresponds to the size of a stride during locomotion.
- A_e defines the amplitude of the extension during swing phase. This corresponds to the ground clearance of the feet during the swing phase.
- θ defines the extension difference between when the leg is at the end of the swing and when the leg is at end of the stance. This is mostly useful for climbing up or down.

We compute t' based on the swing and stance phases:

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

where β defines the proportion of the duration of the swing phase to the stance phase.

For each leg, the phase is calculated separately as

$$\phi_{\text{leg}} = \phi_t + \Delta\phi_{\text{leg}} \bmod 2\pi,$$

where $\Delta\phi_{\text{leg}}$ represents the phase difference of this leg compared to the first (left front) leg. This is defined by the selected gait (i.e. walking vs bounding).

Training

Process

- **Training Algorithm:** Evolutionnary Strategies (ARS) / Reinforcement Learning (PPO)
- Gradually increasing speed from 0 m/s to v_{max} and decrease it to 0
- Add of random perturbations **(vertical up to 60 N/ horizontal up to 10 N)** to favor robustness
- **Policy Complexity**
  - Simple Linear Policy **(ES-Lin)**
  - Two-Layer Fully Connected Network **(PPO)**

Reward Function

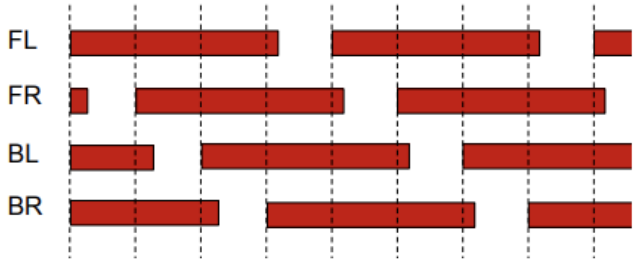
V_{max} : maximum desired velocity
v_R : robot's actual velocity
v_T : robot's target velocity

$$R = v_{\max} e^{-\frac{(v_R - v_T)^2}{2{v_{\max}}^2}}$$

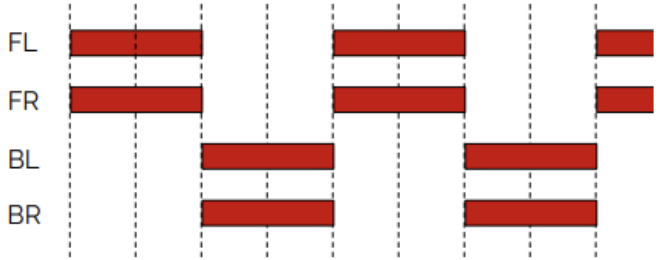
Training

Separate Tasks

- **Slow Walking** (up to 0.4 $m.s^{-1}$)
- **Fast Walking** (up to 0.8 $m.s^{-1}$)
- **Bounding**



(a) Leg phases of TG for walking and running.

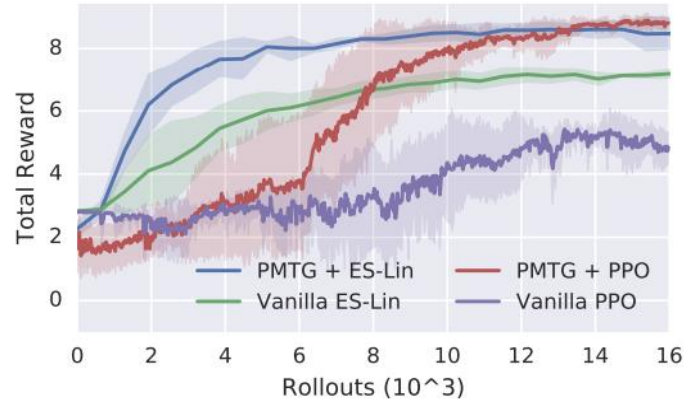


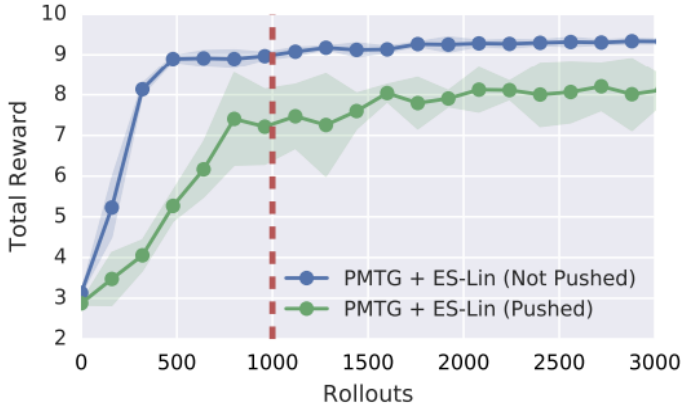
(b) Leg phases of TG for bounding.

Training

Learning with and without PMTG

- Comparison between PMTG and Vanilla (i.e. removal of TG)
  - Better Total Reward in the case of PMTG with both ES and PPO training algorithms.
- Due to the PMTG architecture, even the Linear Policy enables fast and efficient learning.
- Without PMTG, both algorithms failed to achieve optimal reward levels.





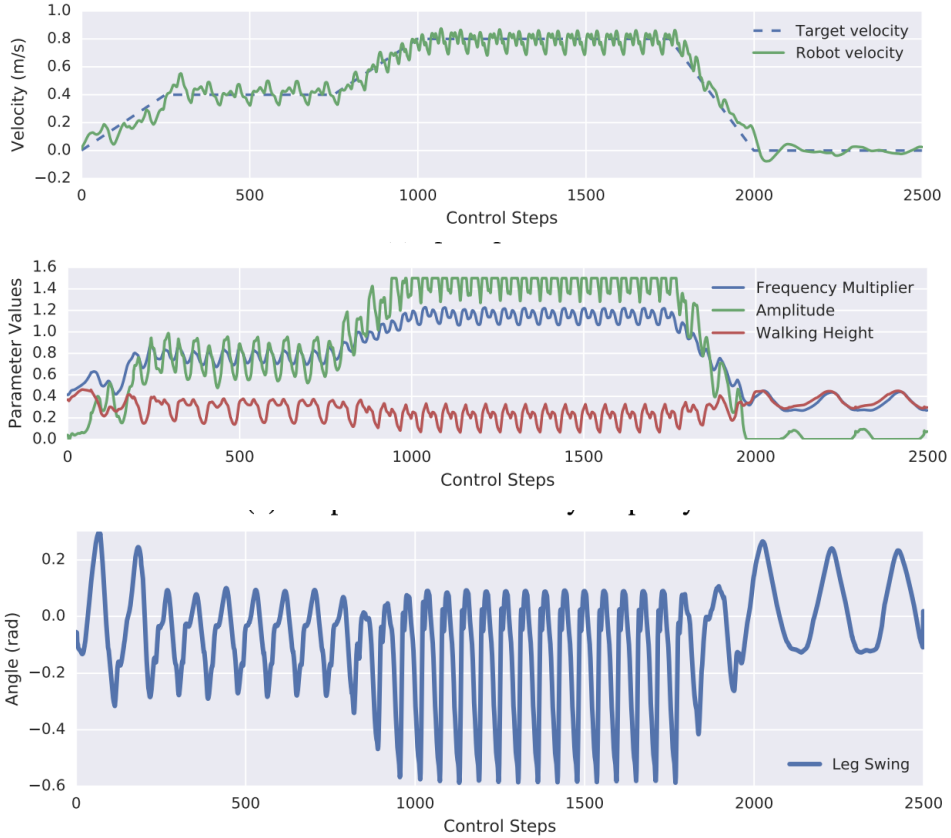
Results

- Single Run after Training
  - The robot can track well the desired speed.

- The policy significantly modulates both the amplitude (which commands stride length) and the frequency of the gait depending on the desired speed.

- It does not necessarily indicate leg movement, as a correction term can be added by the policy.

- The motion of the leg is periodic, but the shape of the signal changes significantly depending on the speed.



Learning Quadrupedal Locomotion over Challenging Terrain (ETHZ)

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, Marco Hutter

- PMTG architecture used to control the ANYmal.
  - Use of temporal convolutional network (TCN) instead of a feed-forward Neural Network in the control policy.
    - Proprioceptive memory for robustness without vision.
  - Training of the policy using a Teacher model that has access to privileged information (contact forces, terrain profile, disturbances, ...).
  - Training on more difficult terrain as the model learns in addition to external disturbances.
  - Robot control uses direction of travel instead of speed.
    - Can adapt the speed depending on the terrain.
    - Omnidirectional.

(More information on the 3rd of December with gr 31)

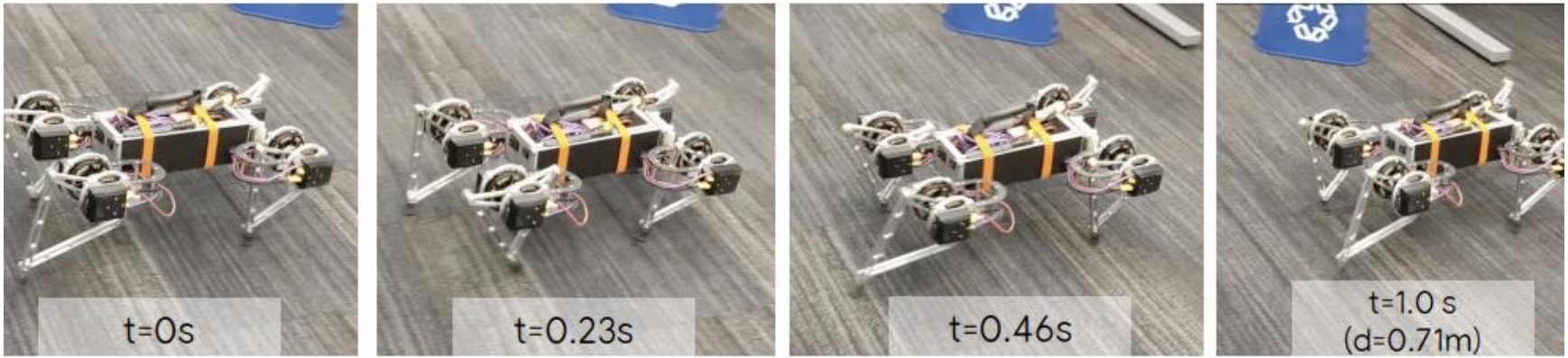
Pros

- Simple architecture
  - 1- or 2-layer Feedforward NN control policy.
  - Motion tracking with PD controller
- Locomotion with walking/running gaits and bounding gaits
- Quick training of the policy.

Cons

- Unidirectional control
- Very simple simulation training, so no adaptation depending on the terrain.
- Blind controller (IMU) is an excellent basis for locomotion, but it is restricted by the topography (no cliffs, lakes, ...)

Experiments



https://www.youtube.com/watch?v=_6SbXK4i0mY&ab_channel=AtilIscen

Possible exam questions

1) In the context of quadruped locomotion, which parameters of the Trajectory Generator (TG) are adjusted by the policy at each time step?
   - Slide 5
2) Why is the use of a simple linear policy sufficient to learn complex behaviour when using a PMTG?
   - This is because the PMTG reduces the complexity of the learning task by encoding prior knowledge in the trajectory generator (TG), allowing the policy to learn complex behaviour with a simple linear structure. (Slide 9)