Experimental Reproduction Milestone of the Meta-RL for Optimal Design of Legged Robots

[Group 2] Caomeng Zhang, Kai Qiao, Yining Shen, Zhitao Yong, Wangzhuo Fan and Zherui Xu

Abstract—**Following the experimental setup of the paper 'Meta Reinforcement Learning for Optimal Design of Legged Robots', we have achieved numerous milestones in the reproduction process. Upon configuring the simulation environment around Raisim, essential packages including the model of our experiment robot Anymal have been tested successfully. Besides, the application of reinforcement-learning-related methods including Proximal and Policy Optimization (PPO) and Model-Agnostic Meta-Learning (MAML) have been studied relatively thoroughly, in prepare for the next stage for our reproduction project.**

I. Introduction

As arduous designing robots is, the parameters including limbs length, servomotor torque and more are keeping the designing from a clear insight into how these parameters affecting the robots' final performance. With the development of machine learning, a novel approach featuring meta reinforcement learning may utilize computer science to optimize the robotic design parameters.

II. Problem Statement

In the field of legged robotics, due to the large number of design parameters to be tuned, the process of design is often troubled with unclear correlation between the design parameters and the corresponding robot behaviors, resulting in rather sparse principle in designing and model-based methods in computational designing optimization. In conventional paradigm of design optimization, biology inspiration and human approximation are the widely used principle, leading to the ambiguous relation between the parameters and robotic output. While in computational paradigm, design is heavily dependent on model-based approach with over-simplified constraints and results specifying in predetermined tasks.

III. Literature Review

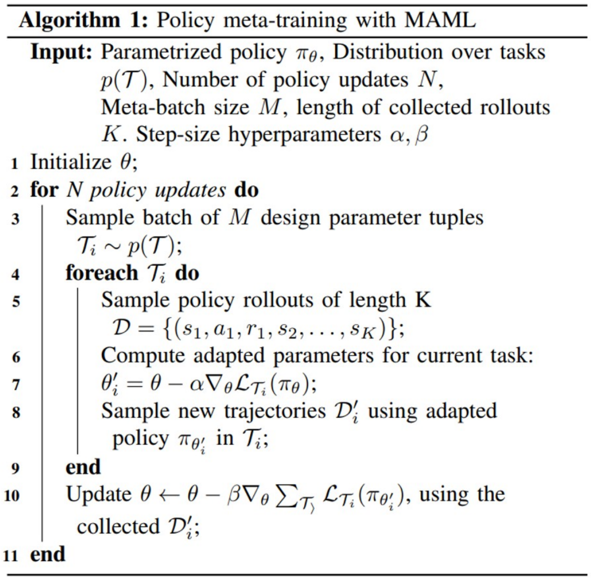
When encountering the insufficiency of training data in machine learning, traditional approaches tend to return unsatisfying results with data overfitting, while the difficulty of acquiring data as training sets exists in most research fields as well as robotics design. In 2017, Finn etc. proposed Model-Agnostic Meta-Learning (MAML) to train a policy with relatively small number of training data [1]. The TensorFlow experiment compares the results to an oracle that receives the identity of the task as an additional input to act as an upper bound on the performance of the model, proving the ability of MAML to do fast learning, to do multiple domains meta-learning, and to keep improving with additional gradient updates or examples.

In the work proposed by Belmonte-Baeza etc., a meta reinforcement learning approach is proven feasible in optimizing the robotic design [2]. Specifically, a robust and adaptive neural network controller framework is equipped with a meta reinforcement learning locomotion control policy, whose experimental results demonstrate the viability.

IV. Technical Approach

In order to exploit the robustness and versatility of reinforcement-learning-based control methods to obtain a generalized locomotion policy, the approach to the demonstration basically consists of two phases: using Meta-RL to train a policy with randomly sampled design parameters and terrains [4], and evaluating different different designs to find the one maximizing design objectives. The processes of training and optimization are separated, considering future utilization of the training results for different optimizing objectives.

With the given design parameters, Markov Decision Process (MDP) is used to model the locomotion control problem in the first place. The MAML is then used to train the meta-policies, ensuring fine-tuning with small amount of data during test time. In the deisgn optimization process, CMA-ES is used for the non-differentiable way of optimizing evaluation.



```
Algorithm 1: Policy meta-training with MAML
Input: Parametrized policy $\pi_\theta$, Distribution over tasks
       $p(\mathcal{T})$, Number of policy updates $N$,
       Meta-batch size $M$, length of collected rollouts
       $K$. Step-size hyperparameters $\alpha, \beta$
Initialize $\theta$;
for $N$ policy updates do
    Sample batch of $M$ design parameter tuples
      $\mathcal{T}_i \sim p(\mathcal{T})$;
    foreach $\mathcal{T}_i$ do
        Sample policy rollouts of length K
          $\mathcal{D} = \{(s_1, a_1, r_1, s_2, \ldots, s_K)\}$;
        Compute adapted parameters for current task:
        $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\pi_\theta)$;
        Sample new trajectories $\mathcal{D}'_i$ using adapted
          policy $\pi_{\theta'_i}$ in $\mathcal{T}_i$;
    end
    Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_j} \mathcal{L}_{\mathcal{T}_i}(\pi_{\theta'_i})$, using the
      collected $\mathcal{D}'_i$;
end
```

Fig. 1. Policy meta-training with MAML

```
Algorithm 2: Design optimization with meta-policy
Input: Trained meta-policy $\pi_{\theta_0}$, Number of
       generations G, Initial design population $\mathcal{P}_0$,
       step-size hyperparameter $\alpha$, number of gradient
       updates $U$, lenght of collected rollouts $T$.
for k in [1...G] do
    foreach $p_i \in \mathcal{P}_k$ do
        Set current design to $p_i$;
        Set policy parameters to initial value: $\theta \leftarrow \theta_0$;
        for U gradient updates do
            Sample policy rollouts of length T
             $\mathcal{D} = \{(s_1, a_1, r_1, s_2, \ldots, s_T)\}$;
            Perform adaptation step:
            $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{p_i}(\pi_\theta)$;
        end
        Compute fitness score for $p_i$ and store it;
    end
    Update $\mathcal{P}$ using the computed scores.
end
```

Fig. 2. Design optimization with meta-policy

V. Preliminary Result

Under the efforts of weeks, we have reached several monumental achievements that would greatly lighten the burden for the next stage of experimental reproduction and show promise in obtaining our own results.

A. MDP Realization in Program

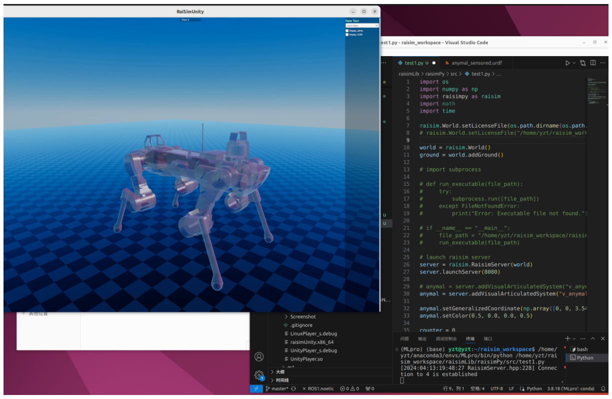
In the context of Anymal locomotion problem, we have built the first version of MDP with its state space $\mathcal{S}$, action space $\mathcal{A}$, transition probability function $\mathcal{P}(s_{t+1}|s_t, a_t)$, and reward function $\mathcal{R}(s_t, a_t, s_{t+1}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ determined. The MDP akin is, as in the paper, defined to be composed as the velocity command, linear and angular body velocity, joint states, frequency and phase of the gait pattern generators for each foot, and the two last actions taken by the policy. Besides, Gaussian policy with Multi Layer Perceptron and Proximal Policy Optimization are used in policy modeling and optimization [3].

B. MAML Test Run

Upon building a simple MAML framework with simple random input, we have successfully executing a MAML program running, as a practical feasibiltiy demonstration for the later MAML running with true Anymal design parameters input. The next step of optimizing the design parameters with MAML is to determine the initial value and varying domain of the parameters and the iteration details, which requires further progress in the experiment setup.

C. Raisim Environment Configuration

Retrieving from the references and Github, we finished the inital configuration of the simulation environment Raisim in Linux Operation System. Additionally, the robot Anymal and Anymal_C together with their major sensors have been presented in the simulator, which will followed by the terrain setup and simulation under optimized design parameters.



```
import os
import numpy as np
import raisimpy as raisim
import math
import time

raisim.World.setLicenseFile(os.path.dirname(os.path.
# raisim.World.setLicenseFile("/home/yzt/raisim_work

world = raisim.World()
ground = world.addGround()

# import subprocess

# def run_executable(file_path):
#     try:
#         subprocess.run([file_path])
#     except FileNotFoundError:
#         print("Error: Executable file not found.")

# if __name__ == "__main__":
#     file_path = "/home/yzt/raisim_workspace/raisin
#     run_executable(file_path)

# launch raisim server
server = raisim.RaisimServer(world)
server.launchServer(8080)

# anymal = server.addVisualArticulatedSystem("v_anym
anymal = server.addVisualArticulatedSystem("v_anymal

anymal.setGeneralizedCoordinate(np.array([0, 0, 3.54
anymal.setColor(0.5, 0.0, 0.0, 0.5)

counter = 0
```

```
(MLpro) (base) yzt@yzt:~/raisim_workspace$ /home/
yzt/anaconda3/envs/MLpro/bin/python /home/yzt/rai
sim_workspace/raisimLib/raisimPy/src/test1.py
[2024:04:13:19:48:27 RaisimServer.hpp:228] Connec
tion to 4 is established
```

Fig. 3. Raisim simulator setup

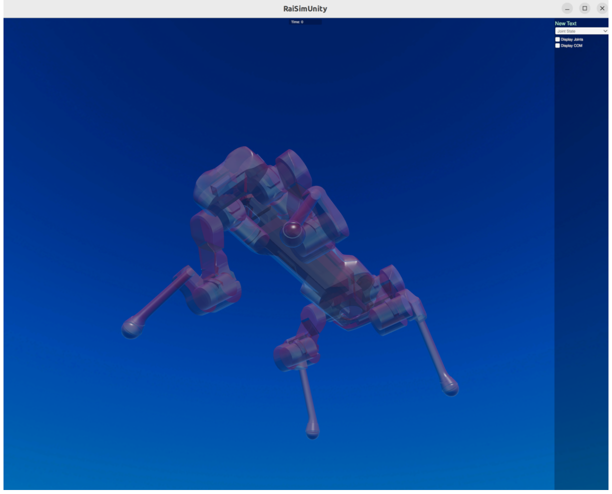


Fig. 4. Anymal visualization in Raisim

VI. Conclusion

The application of Meta-RL into the parameters design in legged robotics has been demonstrated to be promising in solving the conventional difficulties including sparse design principle and model-constraint computational optimization. We have reached the initial stage plan of the experimental reproduction, finishing the preliminary mathematical and programming setup, as MDP realization and MAML testing in an early stage, which lays important foundation for the later work of doing the direct optimization and simulation with Anymal in Raisim simulator. The work ahead includes specified application of the mathematical frameworks parameters onto the Anymal locomotion control policy, and detailed controlling commands of Raisim. The total work is set to be finished by the end of the semester, producing the simulation of the optimal design parameters and performance comparison between different setup.

References

[1] Chelsea Finn. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *International conference on machine learning*, pp. 1126-1135. PMLR, 2017.

[2] Álvaro Belmonte-Baeza, et al. Meta Reinforcement Learning for Optimal Design of Legged Robots. *IEEE Robotics and Automation Letters*, 7(4):12134-12141, 2022.

[3] J. Schulman et al. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017

[4] J. Lee et al. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, vol. 5, no. 47, 2020. [Online]. Available: https://robotics.sciencemag.org/content/5/47/eabc5986