Full Stack GPU Training and An Architecture of RL Policy with Optimization-Based Control

Xingye Da
Nvidia Corporation
Santa Clara, CA, USA
xda@nvidia.com

Zhaoming Xie
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada
zxie47@cs.ubc.ca

I. Background

The GPU has been widely used in machine learning. For a basic RL pipeline, the GPU is usually used to compute the neural network, while the CPU is used for simulation. The bottleneck of the training speed is from the CPU part, either the simulation speed or the limited threads for parallelization. Even we improve the CPU performance, the data transfer between CPU and GPU will dominate the computation. A full-stack GPU training pipeline mitigates the data transfer issue and allows massive parallelization. Based on our benchmark, this new GPU training pipeline can be 10-1000 times faster than the classic GPU+CPU pipeline.

Another performance issue is related to the optimization-based control. When an RL environment includes an optimization-based controller, the time to solve a QP or NLP dominates the computation. Our full-stack GPU pipeline reduces the overall training time by massive parallelization. We can run thousands of environments on a single GPU compared to tens in a CPU. We also show an architecture that an instant QP with RL policy performs equivalent to an NMPC but computes in a fraction of time.

II. Methods

A. Full Stack GPU pipeline

We use Nvidia Isaac Gym to build the full stack GPU pipeline. The Isaac Gym enables simulation on GPU and returns PyTorch tensor. The tensor is passed to a neural network and to compute rewards. The tensor includes states for all robots/environments, and no OpenMP, MPI are required. During the training, all data is stored in the PyTorch GPU tensor. Thus, no data transfer between CPU and GPU is required. The pipeline can be created all in Python.

B. RL Policy with Optimization-based Control

We created an architecture that combines a high-level RL policy and a low-level optimization-based controller. The RL policy takes robot states, local height map, and user command as input and outputs center of mass target acceleration, contact phase speed, and target feet locations for the low-level controller. The low-level controller is composed of a body controller and a swing leg controller. The body controller converts the center of mass target acceleration to ground reaction force by solving a QP. The swing leg controller converts the target feet' locations to trajectories and then tracks the trajectories. The contact phase decides the stance/swing switches.

We have written a QP solver on GPU for our specific problem. Thus, we can run the entire feedback system/environment on GPU. Moreover, even our QP solves a problem at the current instant. We show the overall controller can be equivalent to a nonlinear model predictive control (NMPC) because the RL cumulates the reward and learn to predict the future implicitly.

III. Results & Discussion

We test our setup on a quadruped Laikago. We create a challenging terrain of stepping stone with random height. We compare our learning results with a high-level heuristic controller. When the task is more difficult, the heuristic controller will fail, and the RL-based controller can complete it until it reaches some physical limits. The training currently tasks a few hours, but we are working on getting it done within an hour by training in the cloud.

IV. Conclusion

We provide a full-stack GPU training pipeline that reduces the training time by mitigating the data transfer between GPU and CPU. The capability of massive parallelization suites well for the environment includes an optimization-based controller since the optimization is typically slow. Moreover, we show an instant QP + RL policy has a similar performance to NMPC but runs in a fraction of time.