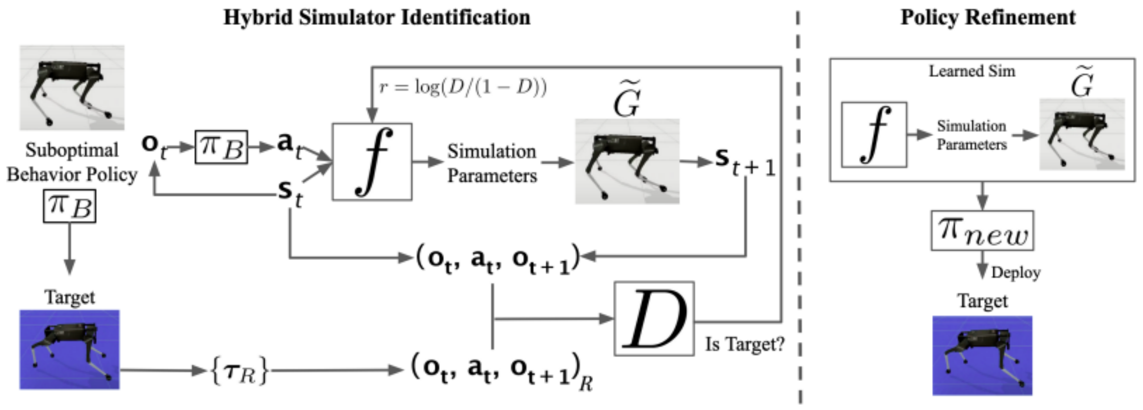
SimGAN: Hybrid Simulator Identification for Domain Adaptation via Adversarial Reinforcement Learning

Introduction and Motivation

There is a significant difference in the performance of robots in Software physics simulators and the real world. It is caused by varying abstractions and approximations of real world phenomena like friction, contacts, and various models of forces. Reducing this "Sim-to-real" gap is the primary purpose of this paper. We learn a Hybrid simulator that can balance model expressiveness and maximizing the state space where the model is valid. The solution approach is described below.

Solution Approach

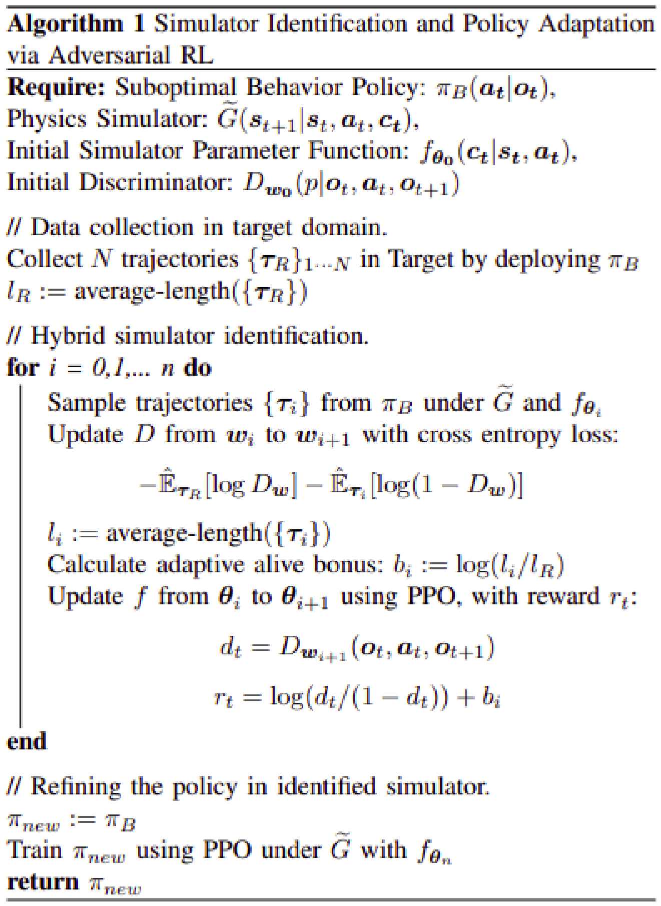


The first step is the Hybrid Simulator Identification.
We have a Target domain and a simulation domain. The target domain can itself be a simulation environment that emulates real world conditions and can induce a "sim-to-real gap".
We assume that any dynamics can be modeled with a function f that is the physics model of the target domain with appropriate parameters. For example, it could include motor torques, friction and contacts. This is the function we need to learn from the Target domain in the simulation domain. The simulation domain f is to be improved to match the Target domain f.
We generate a small number of trajectories in the Target domain (TauTarget). We fix a Behavior policy (PiB) and use that to generate trajectories in the simulation domain as well. To find the divergence/difference in the generated trajectories in the two domains, we use a GAN discriminator instead of the usual MSE loss, as this eliminates the need to abstract the trajectory loss to a distribution loss. It automatically induces trajectory matching on the distribution level due to its nature.
To minimize the GAN loss on the trajectory tuples, we treat the Simulator parameter function f as the RL agent and use Policy Gradient. Also, like in a typical GAN framework, the training of the Discriminator D_w alternates with the training of the Generator f. The reward for the training of f is dependent on the discriminator output D itself. D is given as 0 when the trajectory is

identified as a simulation perfectly, and 1 when it is identified as a real/target domain perfectly, otherwise in between. Thus, at the end of the first step, we have learned a function f that perfectly models the simulation parameter function of the target domain.

The second step is the Improved Policy generation in learned Hybrid simulator
Once we have a learned hybrid simulator, we use that hybrid simulator to train a better policy (PiNew) for the target domain (which has now been accurately modeled in the simulation domain) using Deep reinforcement learning. We can directly test this new and improved policy directly on the Target domain.

The Complete Algorithm is given here $\longrightarrow$



```
Algorithm 1 Simulator Identification and Policy Adaptation via Adversarial RL
Require: Suboptimal Behavior Policy: $\pi_B(\boldsymbol{a_t}|\boldsymbol{o_t})$,
Physics Simulator: $\widetilde{G}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{c_t})$,
Initial Simulator Parameter Function: $f_{\boldsymbol{\theta}_0}(\boldsymbol{c_t}|\boldsymbol{s_t}, \boldsymbol{a_t})$,
Initial Discriminator: $D_{\boldsymbol{w}_0}(p|\boldsymbol{o}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1})$
// Data collection in target domain.
Collect $N$ trajectories $\{\boldsymbol{\tau}_R\}_{1\ldots N}$ in Target by deploying $\pi_B$
$l_R :=$ average-length$(\{\boldsymbol{\tau}_R\})$
// Hybrid simulator identification.
for $i = 0, 1, \ldots\ n$ do
    Sample trajectories $\{\boldsymbol{\tau}_i\}$ from $\pi_B$ under $\widetilde{G}$ and $f_{\boldsymbol{\theta}_i}$
    Update $D$ from $\boldsymbol{w}_i$ to $\boldsymbol{w}_{i+1}$ with cross entropy loss:
        $-\hat{\mathbb{E}}_{\boldsymbol{\tau}_R}[\log D_{\boldsymbol{w}}] - \hat{\mathbb{E}}_{\boldsymbol{\tau}_i}[\log(1 - D_{\boldsymbol{w}})]$
    $l_i :=$ average-length$(\{\boldsymbol{\tau}_i\})$
    Calculate adaptive alive bonus: $b_i := \log(l_i/l_R)$
    Update $f$ from $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_{i+1}$ using PPO, with reward $r_t$:
        $d_t = D_{\boldsymbol{w}_{i+1}}(\boldsymbol{o}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1})$
        $r_t = \log(d_t/(1 - d_t)) + b_i$
end
// Refining the policy in identified simulator.
$\pi_{new} := \pi_B$
Train $\pi_{new}$ using PPO under $\widetilde{G}$ with $f_{\boldsymbol{\theta}_n}$
return $\pi_{new}$
```

Some other important things to keep in mind :

- The number of trajectories to be sampled from the target domain is quite small, and sufficient. Apart from the initial generation of them, no further “training trajectories” from the target domain are needed.

- The similarity reward required to train f to generate trajectories similar to target domain is provided by the colearned discriminator itself. It is based on the Cross Entropy loss including terms for both the target domain as well as the simulator domain.

- The goal of the first step (Hybrid SysID) is to learn the simulator parameter function f. It can also be modeled as a neural network.

- The behavior policy (PiB) does not change in the Hybrid Simulator SysID step.

- We include a minor addition to the reward function for training f, called the Adaptive alive bonus, that is proportional to the ratio of the length of the trajectories in the Simulator and the Target domain. This constrains the trajectories from being too short (over early termination).