[4.1.2021]

[Sim-to-Real: Learning Agile Locomotion for Quadruped Robots]

Summary

This paper tries to handle the issue of the simulation-reality gap. Some common problems of the simulation environment and how they are handled in this paper.

- Simulation environment has a gap between the true robot model → system identification, more accurate model, consider latency, etc. → make the simulation model more realistic.
    - Can latency be learned if we add more historical data into the black box?
- However, the model cannot be perfect, and some errors are inevitable, wear and tear, unmodeled dynamics, we need robust controller, to achieve that in simulation
    - add randomness to parameters, at the beginning of episode, introduce some randomness to parameters, and the random extent depends on how reliable the parameter is.
        - One result with randomness is that it can help decrease the gap between simulation and experiments, but the performance in simulation alone is reduced → interesting result.

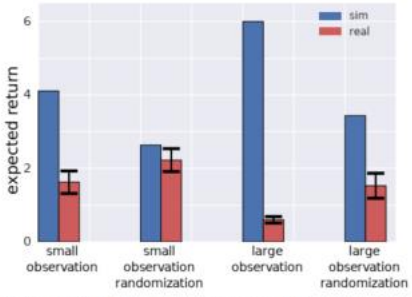

Fig. 9: Comparison of controllers trained with different observation spaces and randomization. The blue and red bars are the performance in simulation and in the real world respectively. Error bars indicate one standard error.

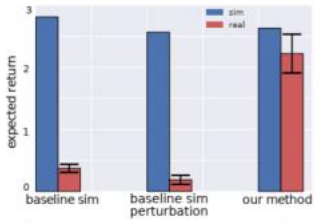

Fig. 6: Controller performance in simulation (blue) and on the robot (red). From left to right, the controllers are trained using baseline simulation, using baseline simulation with random perturbations, and using improved simulation with random

        - Two ways of avoiding overfitting from this picture:
            - Compact observation
            - Add randomization
            - Perturbations
    - Use a more compact observation space will also help. Can avoid overfitting and generalize better normally.
    - Add perturbation (lateral force) in simulation
- The most important idea of this paper, probably the first paper that puts this idea: simply learning everything from the scratch is not working well. Need some essential parameters, I prefer to call it prior knowledge that is difficult/unnecessary for the network to learn that make it work better.
    - In this paper, the idea of gait generator is proposed, based on some parameters (fixed, therefore cannot adjust to changing speed target), and time phase, the trajectory can be roughly determined → as a guidance, or part of the policy.

Comparison

- a later paper uses the basic same idea and framework, except that paper learns some parameter as well for generating the trajectory.
    - In this paper, the way of trajectory being generated is fixed and defined by two parameters: swing signal and extension signal → such as way is determined by the way the robots are built. → however, the idea can be leveraged.

. The signal consists of two sine curves.

$$\bar{s}(t) = 0.3\sin(4\pi t)$$
$$\bar{e}(t) = 0.35\sin(4\pi t) + 2$$

- Detail about the network, interface wise:
    - Action space: desired pose of each leg → position control for some reasons (need to check the paper elaborate more on that), some mapping is required but rather technical.
    - Observation space: roll, pitch, angular velocities of the base along these two axes.

        - Yaw is not included, because too noisy
        - Other noisy data is not included as well
        - A compact observation helps better learning performance.
    - Choice of gap measurement KPI, expected return → the return is defined as 1. If the robot is going in the right direction 2. Energy consumption is penalized.
- The analysis in another papers says, this paper is using a direct-drive actuator → easier than electrical actuator (as used in ANYmal, Cheetah) to control.

Thoughts

- Need to check something about POMDP, state space and observation space, almost forgot….