[4.1.2021]

[Learning Agile and Dynamic Motor Skills for Legged Robots]

Summary

To control the robots, one popular approach is called modular controller design, the control problem is split into three/two subproblems:

1. Old fashion method: first modules compute the desired foot hold positions, second modules compute the trajectory based on the foothold positions, the third module tracks the trajectory.
    a. Limitations: accuracy issue, laborious parameter identification work
2. More recently: trajectory optimization is used to integrate the first two module.
    a. Main issue is numerical optimization results in sub-optimal
    b. The optimization is built when the robot is moving → either in form of motion planning or MPC → lots of computation burden → want a direct mapping, that would be so cool!!

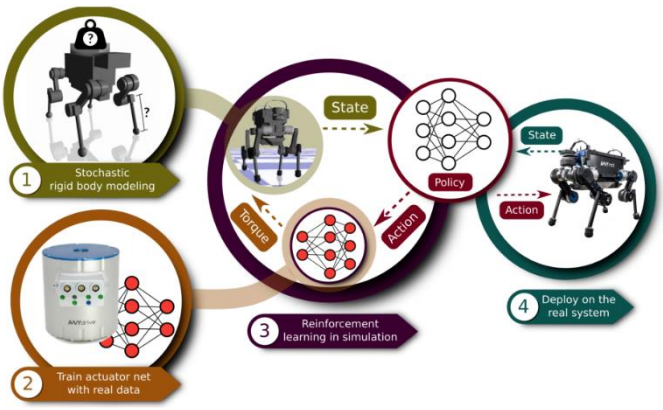

Fig. 1. Creating a control policy. In the first step, we identify the physical parameters of the robot and estimate uncertainties in the identification. In the second step, we train an actuator net that models complex actuator/software dynamics. In the third step, we train a control policy using the models produced in the first two steps. In the fourth step, we deploy the trained policy directly on the physical system.

To handle the issues mentioned above, data-driven reinforcement learning is used:

1. In what level do we use neural networks? → the actuator behavior is learned using neural network → input: history information about joint position error and velocity error, that should be sufficient to compute the next torque prediction (acceleration), the sampling frequency should be high, history should not be too long, in case of some contact will tremendously change the joint status.
2. After the actuator network is learned, then it will be plug into the policy network and reinforcement learning is used to train the action (torque) directly → in simulation only.

- Loss function is the key factor in training a successful policy network. For different tasks, different loss function has to be determined manually by tuning parameters and adding some intuition loss
    - One argument of the paper is that the time spent on parameter tuning is still much less then system identification, etc. if we use a traditional control approach.

Comparison

- Actuator difference results in different algorithms:
    - Boston dynamics uses hydraulic actuator, Cheetah and ANYmal both uses electrical actuator. Real-To-Sim paper uses a direct-drive actuator
        - Electrical actuator's behavior is more complicated→ difficult to analytically model the mapping directly.
- Learning objective: In a later ANYmal learning paper, the network policy's action space consists of foot position residual and leg frequency → will pass to Trajectory Generator → might be more suitable for a challenging terrain as leg frequency need to be changed from time to time. In the original paper of TG, the second neural network is preserved as well.

- In this paper, the action space of the policy network (the main network) is target joint position → looks like a little bit outdated.

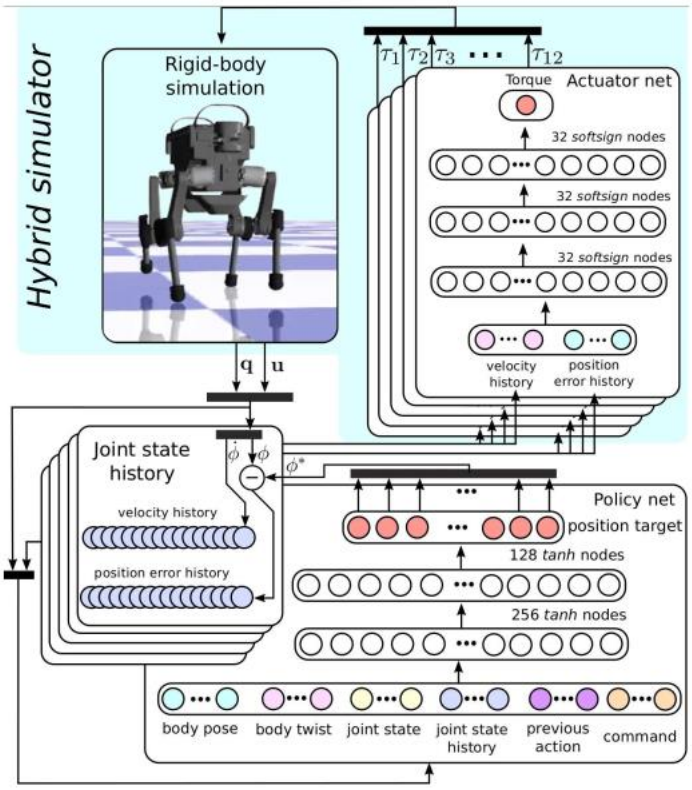

Fig. 5. Training control policies in simulation. The policy network maps the current observation and the joint state history to the joint position targets. The actuator network maps the joint state history to the joint torque, which is used in rigid-body simulation. The state of the robot consists of the generalized coordinate **q** and the generalized velocity **u**. The state of a joint consists of the joint velocity $\dot{\phi}$ and the joint position error, which is the current position ϕ subtracted from the joint position target ϕ^*.

- However, the target position here is not serving for the torque control directly. Because torque/actuator has complex behavior considering model inaccuracy and delays, etc. The torque is generated by a network and deployed on the robot in simulation environment→
-

Thoughts

- Depending on the reinforcement is used only in simulation or in simulation and experiments→ torque generation can simply be done by network in simulation (training data is obtained from the real experiment because we need the actual behavior)
    - Torque/actuator behavior is not relevant to the actual task? Because the training data is obtained with normal actions?
    - I guess this network is trying to simply get the “analytical mapping” from the joint level information to torque command → a different comparison would be to made with analytical solution, for example, with system identification approaches.
- Thoughts 2