Investigating Overconstrained Quadruped Locomotion using Reinforcement Learning

Guojing Huang, Jinda Dong, Junwei Lu, Xinyan Ju, Zhou Chen, Shengyang Ming

I. Introduction

What we want to study is to train a quadrupedal robot through (deep) reinforcement learning to pass all kinds of obstacles on its own, such as "high hurdles", "slopes", "steps", "double wooden bridges", and "stumps". This problem is interesting because quadrupedal robots are becoming more and more popular in scientific research and commercialization, and reinforcement learning is a learning control algorithm that has been widely used by research institutes and commercial companies in recent years, so we would like to combine both of them for this course project.

II. Problem Statement

For the problem we investigated, we propose a design and learning method to tarin and verify the capabilities of overconstrained locomotion in challenging environments including high hurdles, double wooden bridges and stumps,using reinforcement learning. We expect the result that self-design overconstrained quadruped can have great performance in complex environments with RL-trained control policy.

We have now collected some preliminary data, one is the robot ontology configuration, firstly, we have collected the stl model of the robot as well as the available urdf and usd files for describing and simulating the robot; secondly, we have chosen the Cybergear, as the driver of the quadruped robot, and we have also obtained the key parameters of the motor based on the instruction manual; then we have chosen to train the quadruped robot in two kinds of simulation environments for training, one is flat ground and the other is self-built field; lastly, the camera, if we need to train on the self-built field, we need to add a camera to the quadruped robot to detect obstacles. Meanwhile, we will obtain some data and support that may be needed in the future by looking up papers, open source materials, product manuals, etc.

We will use "Isaac Sim" developed by NVIDIA as software platform, and the orbit framework is used for training. The training method is rsl_rl, which is a fast and simple implementation of RL algorithm, designed to run fully on GPU. We will validate its feasibility at first and then modify the parameters of learning and control strategy to achieve better performances.

As for results, we currently choose the relatively easy-to-obtain parameters of velocity, COT, reward, and joint torque as the technical indexes for evaluating the training results, which will eventually be analyzed by figures or tables. The training framework orbit offers 3 types of way for us to train our quadrupeds: rsl_rl, sb3, and skrl. If time permitted, we will compare the output performance of these three methods, with same rewards and configs or not. Besides, we have obtained two kinds of quadruped robots, unitree a1 and overconstrained quadruped with Cybergears, we will also try to compare their capabilities using rsl_rl to train.

III. Literature Review

With high-efficiency and high-accuracy, Kalman filter is widely used for estimating the dynamic state of the system from noisy data. (1) states a formation of a certain Kalman filter:

$$x_k = F_k x_{k-1} + B_k u_k + \omega_k \tag{1}$$

where F_k is the state transition matrix, B_k is the control input matrix, u_k is the control vector, and ω_k is the process noise.[4]

Based on the property of such a useful tool, our experiment aims to better the performance of the pose of our model by lowering the effects from the noise, and to better the controlling result of the locomotion by predicting the dynamic parameters through previous collections.

Several researches on bipedal robots (two-legged machines for which walking is a natural dynamic mode) have come into realize. The bipedal robots have higher requirements on complex controlling methods and strategies to provide a relatively stable controlling basement, like keeping balance, resistance to external interference, etc. Reinforcement learning allows a better result on robust, adaption, and model-free controlling [1]. And in real life world, the robust has been proven to be realistic through reinforcement learning [2].

So as to quadruped robots. Some researchers claim that, the use of external perception has always been a great challenge in quadruped robot movement [5]. To address this problem, reinforcement learning also provides strategies by applying hierarchical learning framework and reward function, to decompose the complex locomotion tasks.[3] This method leads to an efficient training process.

For our work, we are going to apply reinforced methods that are similar to those of bipedal robot control on the strategies of controlling our quadruped model. Using such methods allow us to implement the self-adaption, robotic motion navigation, collaboration etc under an unknown circumstance.

IV. Technical Approach

Overall, we need to train the kinematic gait of an overconstrained quadruped robot using reinforcement learning to obtain the best kinematic performance in a given environment with a defined robot configuration as well as actuator and

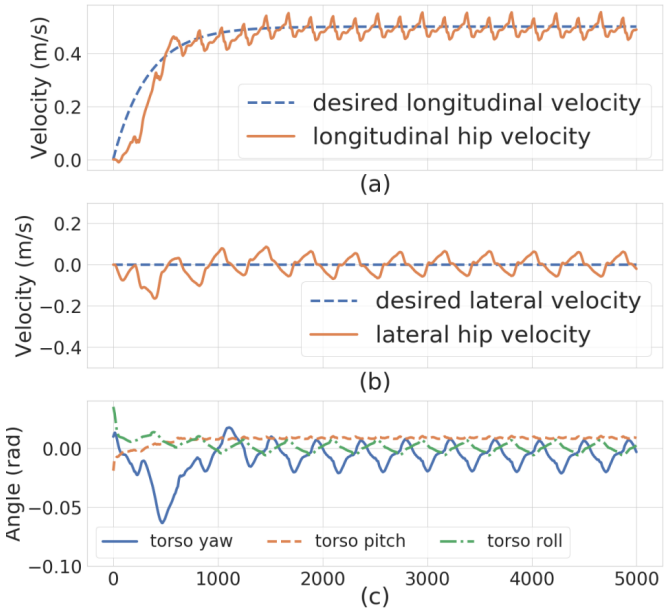


Fig. 1: Simulation on applying reinforcement learning on the bipedal robots. The graphs shows the performance of the learned policy while tracking a fixed desired longitudinal walking speed.[1]

sensor configurations. We will use the reinforcement learning framework rsl_rl for training and custom-design the corresponding robot ontology, PPO solver, and configures for the training environment.

A. *train.py*

This code utilizes the rsl_rl framework for training. Firstly, it uses gymnasium to create an isaac environment, and uses the render mode of rgb_array, and provides a wrapper interface to record the training video. Then, it uses rsl_rl to set up an on-policy training agent, runner, and at the same time, it sets the seed of the training environment to the seed of the agent, so that the initial state of the quadruped robot is the same for each training to obtain reproducible and comparable training results. Finally, it starts the training process through the runner. When entering the training loop, the runner will automatically update the total number of iterations and make the agent start interacting with the environment, and then the agent will select the corresponding action according to the observation. Moreover, the agent will select the corresponding action according to the observation, and after executing the corresponding action, the agent will directly obtain the observation, reward, dones and info of the next moment, and finally the runner will calculate the return of the current training step according to the set discount factor and lambda parameter, which completes the whole closed loop of training.

B. *BENNETTQUAD_A1_CFG*

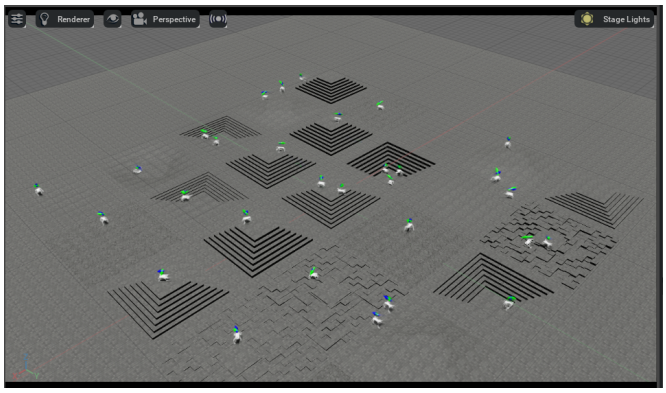
This configuration is based on ArticulationCfg and contains three main features: spawn, init_state and actuators. In spawn, it mainly sets the physical properties of the robot, imports the corresponding usd file, initializes the mass, linear_damping, angular_damping, max_linear_velocity, max_angular_velocity, max_depenetration_velocity, and at the same time initializes the robot's mass, linear_damping, angular_damping, max_linear_velocity, max_angular_velocity, and max_depenetration_velocity. depenetration_velocity are initialized, and both the solver position and velocity iterations are set to 4; the initial state of the robot at the beginning of each training session is mainly set in init_state, including the spatial position, joint position, and joint velocity. We set the spatial position to be the zero point of the absolute coordinate system, the initial joint velocity to be 0, and the joint position to be close to the low lying state; in actuators, we mainly set the mounting joint position of the motor of Yushu A1, as well as the corresponding parameters of the motor body, and we installed motors in the motion joints of the four legs, which correspond to the three degrees of freedom of the legs rotating around the hip joints, the legs swinging back and forth and the foot-configuration transformation. We installed motors in all four legs, corresponding to leg rotation around the hip joint, leg back-and-forth oscillation, and foot configuration transformation with three degrees of freedom and a total of 12 motors, and set the effort_limit and saturation_effort to 10.5, the speed limit to 21m/s, the stiffness to 25.0, the damping to 0.5, and the friction to 0.2.

C. *BennettquadA1RoughPPORunnerCfg*

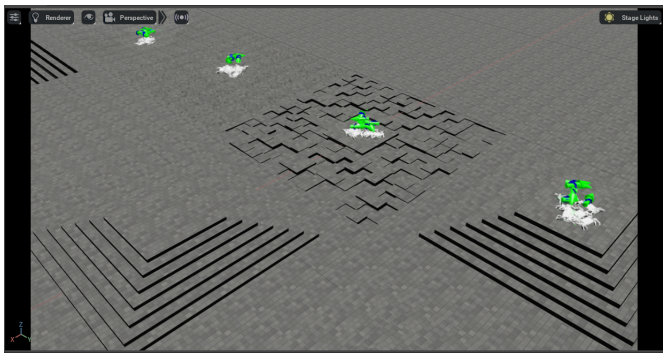
This configuration is inherited from RslRlOnPolicyRunnerCfg and is used to configure the runtime parameters of an RL (Reinforcement Learning) agent based on the PPO algorithm (Proximal Policy Optimization). Specifically, this configuration defines the following parameters: number of steps per environment is 24,maximum number of iterations is 2000, interval to save the model is 50 (the model is saved every 50 iterations). while on the configuration of the policy policy network, the PPO actor-critic networks generation control policy configured through rsl_rl, the initial noise standard deviation of the actor is set to 1.0, and the maximum number of iterations is set to 2,000, the hidden layer dimension of both actor network and critic network is 3 layers, and the number of neurons in each layer is 512, 256, and 128, respectively. The activation function is ELU function. ELU does not have neuron death problem, and it can shorten the training time and improve the accuracy in neural network. And the algorithm also uses the configuration of rsl_rl's PPO algorithm, including the value function loss coefficients, whether to use truncated value function loss, truncation parameters, entropy coefficients, the number of learning cycles, the number of small batches, the learning rate, the scheduling method, the discount factor, the lambda parameter, the expected KL dispersion, the gradient trimming threshold and other parameters.

V. Intermediate Results

In terms of platform setup, we have downloaded the Isaac Sim software and have fully configured the Orbit framework. In the learning process, our group has learned how to create an



(a) Simulation of wheeled-leg version



(b) Simulation of tip-leg version

Fig. 2: Intermediate results of the project

RL environment in the Orbit framework and train a quadruped robot's gait in this environment. As for the results, we have obtained the correct robot model and its URDF and USD files, with two types of legs: wheeled legs and tip legs. We have used SolidWorks to build a complex simulation environment, and we are currently researching how to convert our self-built site into a usable training environment. This involves importing mesh and USD files, setting up virtual boundaries for the site, and other issues. The wheeled-leg version of the robot has been trained to a relatively good state of motion, able to adapt to default complex environments (such as flat ground, rubble, and steps) to maintain a reasonable posture. However, training for the tip-leg robot has not been successful. The current training results show that the tip-leg robot cannot stand, or it collapses after standing for a few seconds due to some reason.

References

[1] Wei Zhang Ayonga Hereid Guillermo A. Castillo, Bowen Weng. Hybrid zero dynamics inspired feedback control policy design for 3d bipedal locomotion using reinforcement learning. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[2] Wei Zhang Ayonga Hereid Guillermo A. Castillo1, Bowen Weng. Robust feedback motion policy design using reinforcement learning on a 3d digit bipedal robot. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[3] Deepali Jain, Atil Iscen, and Ken Caluwaerts. Hierarchical reinforcement learning for quadruped locomotion. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7551–7557, 2019. doi: 10.1109/IROS40897.2019.8967913.

[4] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82 (1):35–45, 1960.

[5] Jemin Hwangbo Lorenz Wellhausen Vladlen Koltun Marco Hutter Takahiro Miki1, Joonho Lee. Learning robust perceptive locomotion for quadrupedal robots in the wild. *SCIENCE ROBOTICS*, 2022. doi: 10.1126/scirobotics.abk2822.