Supplementary Material: Robot Parkour Learning

Anonymous Author(s)
Affiliation
Address
email

1 Experiment Videos

We perform thorough real-world analysis of our system. Indoor and outdoor experiment videos can be found at https://agile-locomotion.github.io.

2 Details of Training in Simulation

The specialize skill policy consists of a GRU followed by an MLP that outputs the target joint positions. We concatenate all the observations including privileged environment information as a flattened vector. It is passed to a one-layer GRU of 256 hidden sizes, followed by an MLP of hidden dimensions as [512, 256, 128]. We use ELU as an activation layer inside the policy. The final layer outputs a 12-dimension vector and be fed to tanh activation function. The action ranges from -1 to 1, which is scaled by a preset constant action scale. The detailed parameters of running PPO algorithm are listed in Table 1.

The obstacles for climbing is 0.8m wide and 0.8m long (along the forward direction). The obstacles for leap are gaps of 0.8m wide and 0.8m depth. For crawling, the obstacle is 0.8m wide and 0.3m along the forward direction. For tilting obstacles, the length along the forward direction is 0.6m.

Table 1: Training Hyper-parameters

PPO clip range	0.2
GAE λ	0.95
Learning rate	1e-4
Reward discount factor	0.99
Minimum policy std	0.2
Number of environments	4096
Number of environment steps per training batch	24
Learning epochs per training batch	5
Number of mini-batches per training batch	4

Hyparameters for training specialized policies using priviledge information. We follow the insights from [1, 2, 3] that use fractal noises to generate terrains, which enforces the foot contact clearance and simplified the reward terms as shown in Table 2. Except for the penetration penalty, we use these parameters to train all 4 specialized policies, in either RL pre-training with soft dynamics constraint or fine-tuning the oracle policies in hard dynamics constraint.

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

Table 2: Reward Scales in training each specialized policy

purpose	hyperparameter	value
x velocity	α_1	1.
y velocity	α_2	1.
angular velocity	α_3	0.1
energy	α_4	$2e-6$
penetration depth	α_5	$4e-3$
penetration volume	α_6	$4e-3$

3 Details of the Parkour Policy Network

The parkour policy is a series of CNN encoders connected with GRU and MLP neural networks. The visual embedding from the visual encoder is concatenated together with the rest of the observation and fed to the GRU and MLP module. The detailed parameters of the network structure are listed in Table 3.

Table 3: Parkour Policy structure

CNN channels	[16, 32, 32]
CNN kernel sizes	[5, 4, 3]
CNN pooling layer	MaxPool
CNN stride	[2, 2, 1]
CNN embeding dims	128
RNN type	GRU
RNN layers	1
RNN hidden dims	256
MLP hidden sizes	512, 256, 128
MLP activate	ELU

We use binary cross-entropy loss for the parkour policy during distillation. The output of both specialized skills and parkour policy ranges from -1 to 1.

$$D(a^{\text{parkour}}, a^{\text{specialized}}) = \left(\frac{1+a^{\text{specialized}}}{2} \log \frac{1+a^{\text{parkour}}}{2} + \frac{1-a^{\text{specialized}}}{2} \log \frac{1-a^{\text{parkour}}}{2} \right) \times 2,$$

where $a^{\text{specialized}}$ is the action from the corresponding specialized skills, a^{parkour} is the action from the parkour policy.

4 Details of Robot Setup

Robot setup in simulation. We use IssacGym Preview 4 for simulation. We generate a static large terrain map before each training, during the training of specialized policies. The terrain consists of 800 tracks with a 20 by 40 grid. We set the difficulty of each track in a linear curriculum manner. The tracks in the same row have the same difficulty but differ in non-essential configurations. The tracks in each column are connected end to end so that whenever the robot finished the current track, it keeps moving forward (+x direction) to the more difficult track. We train each specialized policy in soft dynamics using one 1 Nvidia 3090 computer for 12 hours and tune it in hard dynamics for 6 hours. For distillation, we use 4 computers, each of which is equipped with 1 Nvidia 3090 GPU, that share the same NFS file system. We use 3 computers for loading the current training model and collecting the parkour policy's trajectory as well as the specialized policy supervision. We use the other one computer to load the latest trajectories and train the parkour policy.

Robot setup in the real world. We use the Unitree A1 equipped for our real-world experiments which is equipped with an onboard Nvidia Jetson NX. The robot has 12 joints. Each joint is equipped with a motor of 33.5Nm instant maximum torque. It also has a built-in Intel RealSense D435 camera in front of the robot using inferred and stereo to provide depth images. We use ROS1 on Ubuntu 18.04 which runs on the onboard Jetson NX. We use a ROS package based on Unitree SDK to send

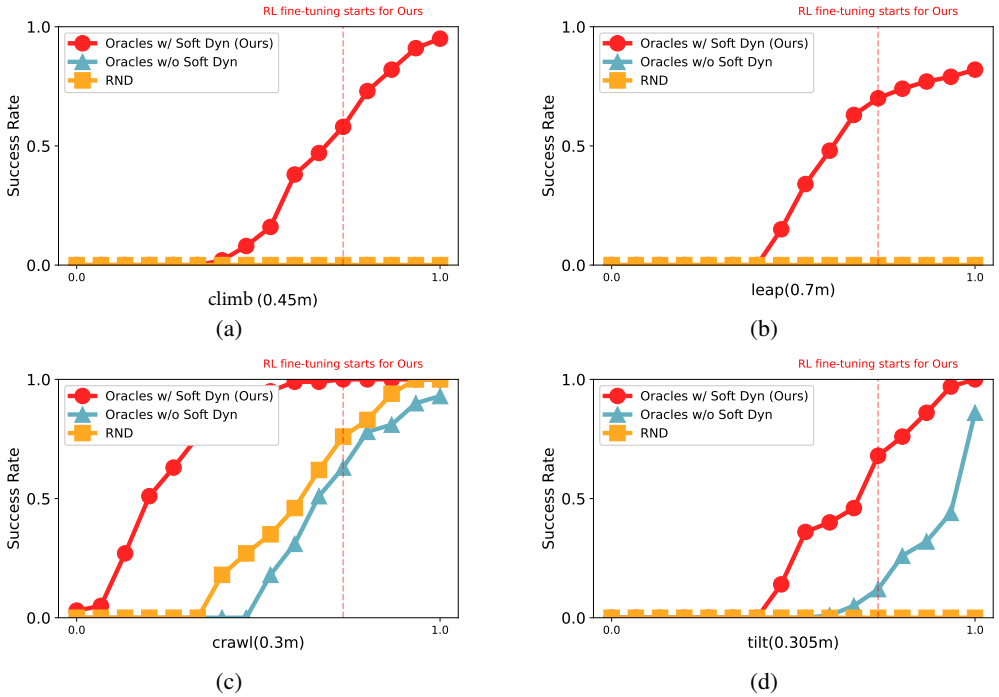

Figure 1: Comparison of our method with Oracles w/o Soft Dyn and RND. For our method, the RL finetuning stage started at the late stages of the training.

and receive the robot states as well as the policy command at 100Hz. The ROS package is also equipped with a roll/pitch limit, estimated torque limit, and emergency stop mechanism using the remote control as the means of protection for the robot. To run the policy, we use two Python scripts to run the visual encoder and the rest of the network asynchronously. We use the Python wrapper of librealsense to capture depth images at the resolution of 240 X 424. We apply the holing filters, spatial filters, and temporal filters from the librealsense utilities. We crop the 60 pixels on the left and 46 pixels on the right before down-sampling the depth image to 48 X 64 resolution. The visual embedding is sent to the rnn script using ROS message at 10Hz. For the script that runs rnn-mlp, we fix the policy frequency to 50Hz. In each loop, we update the robot proprioception and the visual embedding using ROS subscriber and compute the policy output. Then we clip the action by a range computed using the current joint position and velocity at a maximum torque of 25Nm, and send the position control command to the ROS package, with $K_p = 50.0, K_d = 1.0$.

5 Detailed Comparison Studies on RL Pre-Training with Soft Dynamics Constraints

We compare our method with RND and the Oracles w/o Soft Dyn. Our method trained with soft dynamics constraints is the only method that can complete climbing and lepaing skills. As shown in Figure 1 of the supplementary, except for crawling, RND fail to learn successful maneuvers to achieve climbing, leaping, and tilting. Although the Oracles w/o Soft Dyn learned to achieve crawl and tilt skills, the policy trained on climb and leap failed to learn accurate timing to start the maneuver.

References

[1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, Oct. 2020.

[2] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots. In *RSS*, 2021.

[3] Z. Fu, A. Kumar, J. Malik, and D. Pathak. Minimizing energy consumption leads to the emergence of gaits in legged robots. In *CoRL*, 2021.