Learning various locomotion skills from scratch with deep reinforcement learning

D.I. Sorokin* and D.L. Babaev

Autonomous non-profit organization "Artificial intelligence research institute",
Moscow, Russia,
*dmitrii.sorokin@phystech.edu

Abstract. Proficiency in locomotion skills will help robots to navigate over challenging terrains. This task is hard to solve programmatically due to the wide variety of terrains and motion patterns. Here we present a framework to learn an agent capable of solving the task of moving with desired linear and angular velocity. The agent learns the task in a curriculum which gradually increases the difficulty of the learned task. We carefully tune the reward function for the agent. The training process is performed in a simulator with domain randomization which forces the agent to learn a robust policy. We tested the proposed framework on the quadruped robot and achieved competitive results.

Keywords: reinforcement learning, quadrupedal locomotion, curriculum learning

1 Introduction

Robotics is one of the most important applications of reinforcement learning (RL) in real life. Robots that can adapt to changing environmental conditions would have a variety of applications from collaborative robotics [1] to autonomous driving [2]. Reinforcement learning is a promising method to train such robots.

In reinforcement learning, the robot interacts with an environment and learns a policy that would maximize the total discounted reward obtained during an episode. This approach is advantageous compared to a hard-coded policy because an agent can itself figure out an optimal sequence of actions given only a scalar reward signal. Reward as a function of the current state, the next state, and action, is often much simpler to design than the optimal sequence of actions.

However, because an agent interacts with the environment through trial and error, the RL approach has several drawbacks. First of all, the training process requires lots of data for an agent to learn an optimal policy. Second, the reward function should be chosen carefully to prevent the agent from "hacking" it or performing unsafe actions which could damage the robot. Third, if a policy was trained in simulation, it could not generalize well to a real environment due to distribution shift.

Here we present a framework that trains the Unitree A1 [3] robot to solve various locomotion tasks in a simulator. The robot trains in a curriculum which

constantly increases the difficulty of the learned task. We implement a reward function that encourages an agent to learn a safe and smooth locomotion policy. We inject noise into robots' observations in order to learn a more robust policy.

Fig. 1: Unitree A1 robot, "turn counterclockwise" task.

Fig. 2: Unitree A1 robot, "move forward" task.

Fig. 3: Unitree A1 robot, "move backward" task.

2 Related work

The task of legged robot locomotion is a significant challenge in robotics. The successful policy must be robust to observation noise, be able to move safely, and require a small amount of data to train on. One of the most promising approaches to this challenge is based on reinforcement learning. Since not all of the desired quantities can be achieved at once, different works emphasize various characteristics. In work [4] authors present a model-based framework that allows

training of data-efficient policy with only 4.5 minutes of data collected on a quadruped robot which corresponds to 45000 control steps. In work [5] authors train an ANYmal robot [6] using proximal policy algorithm (PPO) [7] with curriculum and domain randomizations, which allowed to deploy the trained policy on the real robot. In [8] authors used two agents, a teacher and a student to walk over different terrains such as stairs or hills. The teacher agent leverages access to privileged information during training while the student agent was trained without privileged information to mimic the policy of the teacher agent. In work [9] authors train rapid locomotion skills using curriculum learning for the MIT Mini Cheetah robot.

The main contribution of our work is as follows: (1) we apply our framework to a new Unitree A1 robot; (2) we propose a reward function that encourages smooth and safe locomotion at the target speed.

3 Background

In reinforcement learning interaction between an agent and an environment is performed sequentially. The agent observes the current state st of the environment and reacts with an action at. The action causes a transition in the environment to the next state s_{t+1}. For the caused transition the agent receives a reward rt. Transitions are assumed to have the Markov property – the future depends on the past through the present which means that the next state of the environment must depend only on the current state and the action. Hence, the behavior policy of the agent should depend only on the current state of the environment.

To train the behavior policy in our work, we use the PPO algorithm [7]. The algorithm uses two neural networks: an Actor and a Critic. The Critic network V approximates the total discounted return:

$$V \rightarrow \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta'}} [G(s_i)] = \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta'}} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

where π_θ - the policy of the agent parameterized by parameters θ; $\mathbb{E}_{\tau \sim \pi_{\theta'}}$ – expectation over trajectories sampled from the current policy $\pi_{\theta'}$; γ - discounting factor, r_t – the reward at the time step t, T – length of the episode. The Actor network $\pi_\theta(\cdot|s_i)$ minimizes the following clipped objective:

$$L = \mathop{\mathbb{E}}_{\tau \sim \pi_k} \left[\sum_{t=0}^{T} \min(\rho_t(\theta) A_t, \text{clip}(\rho_t(\theta), 1-\varepsilon, 1+\varepsilon) A_t) \right]$$

where $\rho_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta'}(a_t|s_t)$ – the likelihood ratio of the action at w.r.t. the current policy and the policy used to collect the data; A_t - generalized advantage estimate [10]; ε - clipping parameter.

The actor's objective is the core of the PPO algorithm [7]. The interplay, between the min operator, likelihood clipping, and the sign-changing multiplicative

advantage A_t enables controllable policy updates which mostly focus on unlikely advantageous or likely detrimental experience.

4 Our method

We carry out our experiments with the Unitree A1 robot. The robot is shown in fig. [1-3]. It is a quadruped robot which has 12 degrees of freedom – positions of the three joints of each of the four legs. We simulate dynamics of the robot using the Raisim simulator [11]. We train and evaluate our agent in the following tasks:

- "Move forward"
- "Move forward with target speed"
- "Move backward"
- "Turn clockwise"
- "Turn counterclockwise"

Observation space has 49 dimensions. They are torso height (1), torso roll and pitch angles (2), joint angles (12), joint velocities (12), previous positions of joints (12), torso linear (3) and angular (3) velocities, indicators of ground contact (4). For the move forward with target speed task observation includes 50th dimension – the target speed. We normalize observation using constant mean and standard deviation.

Actions space is continuous and has 12 dimensions. Actions of our agent are the next positions of robots' joints also normalized by constant mean and standard deviation.

We implement **the reward function** with 9 terms, summed up with coefficients listed in table 1:

$$r = k_{torso\ height} \cdot r_{torso\ height} + k_{torque} \cdot r_{torque} + k_{joint\ speed} \cdot r_{joint\ speed} + k_{slip} \cdot r_{slip} + k_{work} \cdot r_{work} + k_{ground\ impact} \cdot r_{ground\ impact} + k_{z\ acceleration} \cdot r_{z\ acceleration} + k_{velocity} \cdot r_{velocity} + k_{transverse\ and\ rotation} \cdot r_{transverse\ and\ rotation}$$

To specify the terms we use the following nomenclature: k_c - curriculum factor, h_0 – torso height at the beginning of the episode, h – torso height, τ - joint torque, τ_p – previous joint torque, $\|\|$ - L2 norm, V_t - linear velocity of the torso, W - angular velocity, $\hat{\cdot}$ - desired quantity, V_j - linear velocity of joints, V_{ft} - tangential velocity of a foot (x, y components), x_j linear position of joints, x_{pj} previous linear positions of joints, F_f – force between ground and foot, F_{pf} – force between ground and foot at the previous step, D – direction either 1 or -1, subscript x,y,z corresponds to one of the components of the subscripted vector.

The first 7 terms in equation (1) do not depend on a task and are used to motivate the agent to learn a smooth and safe policy. They are defined as follows:

$$r_{torso\ height} = k_c \cdot (h - h_0)^2$$

$$r_{torque} = k_c \cdot \|\tau\|^2$$
$$r_{joint\ speed} = k_c \cdot \|V_j\|^2$$
$$r_{slip} = k_c \cdot (\|V_{ft}\|^2 + W_z^2)$$
$$r_{work} = k_c \cdot |\tau^T \cdot (x_j - x_{pj})|$$
$$r_{ground\ impact} = k_c \cdot \|F_f - F_{pf}\|^2$$
$$r_{z\ acceleration} = k_c \cdot V_z^2$$

For "move forward" and "move backward" tasks $r_{velocity}$ and $r_{transverse\ and\ rotation}$ are defined as follows:

$$r_{velocity} = \text{clip}(V_x \cdot D/\hat{V}_x, 0, 1)$$
$$r_{transverse\ and\ rotation} = k_c \cdot (V_y^2 + W_z^2)$$

For "move forward with target speed" task $r_{velocity}$ and $r_{transverse\ and\ rotation}$ are defined as follows:

$$r_{velocity} = \max(1 - |V_x/\hat{V}_x - 1|, 0)$$
$$r_{transverse\ and\ rotation} = k_c \cdot (V_y^2 + W_z^2)$$

For "turn clockwise" and "turn counterclockwise" $r_{velocity}$ and $R_{transverse\ and\ rotation}$ are defined as follows:

$$r_{velocity} = \text{clip}(W_z \cdot D/\hat{W}_z, 0, 1)$$
$$r_{transverse\ and\ rotation} = k_c \cdot (V_x^2 + V_y^2)$$

To make the learned policy robust we **randomize** the center of mass of the robot in range $COM \sim k_c \cdot U(-0.0015, 0.0015)$, where $U(A, B)$ is a uniform distribution from A to B; friction coefficient $k_f \sim 0.875 + k_c \cdot U(0.5, 1.25)$, motor strength $k_m \sim 1 + k_c \cdot U(0.9, 1.1)$, apply force of 1000 N (which equals to 8.2g for 12.5kg robot) in a random direction to the robot's torso with probability equals to 0.5 at each step and apply random torque of 100 $N \cdot m$ with probability equals to 0.05.

We use a **curriculum** which increases k_c from 0 to 1 during the training process. This lets the agent start with a simple task and gradually adapt to more demanding ones during training.

We use the PPO algorithm as backbone in our framework. To stabilize the learning procedure, we interactively adjust the learning rate using kl-divergence between action distributions from previous and next states:

$$lr_{t+1} = \begin{cases} lr_t/2 & \text{if } kl(\pi_{\theta'}, \pi_\theta) > 2 \cdot kl_{target} \\ lr_t \cdot 2 & \text{if } kl(\pi_{\theta'}, \pi_\theta) < kl_{target}/2 \\ lr_t & \text{otherwise} \end{cases} \quad (1)$$

Table 1: Reward coefficients

$k_{velocity}$	1.25
$k_{lateral\ and\ rotation}$	-0,9
k_{height}	-1,0
k_{torque}	-0,0005
$k_{joint\ speed}$	-0,0015
k_{slip}	-0,1
k_{work}	-0,125
$k_{ground\ impact}$	-0,000015
$k_{z\ acceleration}$	-2,0

Table 2: Parameters of the proximal policy optimization algorithm

clipping parameter	0,2
gamma	0,998
lambda	0,95
value loss coefficient	0,5
entropy coefficient	0,0
learning rate	0,0005
maximum gradient norm	0,5
desired kl divergence	0,01

kl_{target} is target kl-divergence. Maximal length of an episode was set to **3500 steps** which is enough for the agent to reach the bounds of the simulation bounding box by the end of an episode.

We implement both actor and critic using fully connected neural networks with two hidden layers with 512 neurons.

5 Results

We train our agent in 100 simulations running in parallel. Neural networks are updated with backpropagation and Adam optimizer [12] every 128 environment steps. Total number of updates is 5000. We evaluate the trained agents for 100 episodes and present the evaluation results in table 3 and figure 4. Table 3 shows results for the “move forward”, “move backward”, “turn clockwise” and “turn counterclockwise” tasks. It can be seen that in “move forward” and “move backward” tasks agents are more stable and the number of steps is close to the maximal length of the episode. In the “turn” tasks the agents are less stable which results in lower achieved return and lower average number of steps. This can be due to the fact that random forces destabilize the turning agent more than the agent which moves straight.

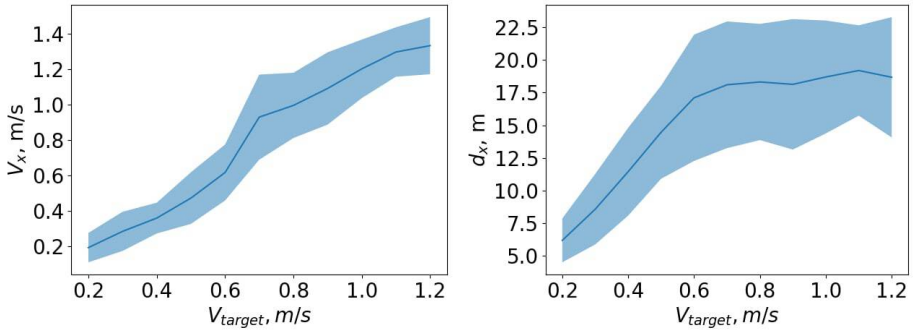

Fig. 4: Evaluation of "Move forward with target speed" task. (a) Achieved speed V_x as a function of target speed V_{target}. (b) Distance traveled by the end of episode as a function of target speed V_{target}.

Fig. 4 shows evaluation results for the "move forward with target speed" task. It can be seen from the Fig. 4a that the agent can move at different speeds and the achieved speed is close to the target. Distance traveled by the end of the episode is shown in Fig. 4b. When speed is between 0 m/s and 0.6 m/s distance grows as the agent goes further. When speed is higher then 0.6 m/s the distance is almost constant and equals distance to the bounds of the environment.

Table 3: Evaluation of trained model in simulation

Task	Move forward	Move backward	Turn clockwise	Turn counterclockwise
Total return	529 ± 124	508 ± 98	85 ± 219	165 ± 155
Number of steps	3263 ± 633	3136 ± 545	2057 ± 1208	2095 ± 1325

Overall, the results show that the Unitree A1 robot can be trained with reinforcement learning and the trained model can achieve good performance.

6 Conclusion

In the present work we developed a framework which can train a Unitree A1 robot to solve the various locomotion tasks. Our designed reward function enforces the agent to learn a smooth policy. Test results show that the trained agent can perform well in a simulated environment.

In the future work we are going to combine all the models into a single agent which can move with arbitrary linear and angular velocities and evaluate the combined model on a physical robot. Afterwards the trained policy can be used

as a first level of hierarchical policy where the second policy would decide where to go and send commands to the first policy. Such agents can learn to solve sophisticated tasks such as moving cargo to a destination.

References

1. S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.
2. B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.
3. https://www.unitree.com/products/a1/.
4. Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," in *Conference on Robot Learning*, pp. 1–10, PMLR, 2020.
5. J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, Jan. 2019.
6. M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 38–44, 2016.
7. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
8. J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986, 2020.
9. G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *arXiv preprint arXiv:2205.02824*, 2022.
10. J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.
11. J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.
12. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.