A Hierarchical Reinforcement Learning Approach to Control Legged Mobile Manipulators

Anonymous Author(s)
Affiliation
Address
`email`

Abstract: Recent years have seen a Cambrian explosion of robotic systems yielding ever more capable and affordable systems, with quadrupedal robotic platforms emerging as a commercially-viable base to perform a wide variety of tasks across uneven terrain. Augmenting these with a robotic arm allows the possibility of even more complex interactions. At the same time, there has been a growing body of research into using deep reinforcement learning (DRL) for embodied agent navigation and object manipulation, which promises a more sample-efficient, flexible, and robust approach to learning such policies than existing classical methods. Recent works have shown a functional approach for learning a joint base and arm policy with DRL but have not yet demonstrated how the result can be used in downstream tasks. In this work, we investigate the problem of learning an object manipulation and navigation policy for a quadrupedal robot with a mounted robotic arm - specifically, we address the problem of fetching stationary and moving objects autonomously ("playing fetch" with the robot dog). Our method consists of (a) a low-level policy that moves the base and arm and (b) a high-level policy that generates the commands for the low-level policy. The low-level policy is jointly learned for both the arm and the base which generates joint torques for directional commands. The high-level policy is task-specific, translates the ball position to directional commands for the low-level policy, and deals with acceleration/deceleration and stability. We demonstrate that our high-level policy can outperform a tuned Proportional-Derivative (PD) controller.

Keywords: Reinforcement Learning, Quadruped Robots, Object Manipulation

1 Introduction

Mobile ground-based robots have been used to solve many tasks such as delivery, inspection, and exploration [1]. More specifically, quadrupedal legged robots have the agility to navigate uneven terrain and traverse obstacles commonly found in indoor settings such as stairs. At the same time, robotic arms are very useful for manipulation tasks that require grasping and picking up objects, but are most often stationary. Merging both of these technologies could yield very effective robotic assistants, capable of accomplishing useful tasks in uncontrolled real-world environment. However, this combination of quadruped base with robotic arm poses a difficult control problem.

Many classic control and Reinforcement Learning (RL) methods have been proposed for each of these domains, quadrupedal locomotion [2, 3] and robotic arm manipulation [4, 5]. Far fewer methods have been proposed for the control of quadrupedal robots with robotic arms. Such methods must take into account the interactions between the two robotic systems, especially when dealing with moving objects or rapidly changing commands. As an example, [6] propose to add a correction term to their state estimation pipeline to compensate for the fact that the arm is moving and affecting the locomotion controller behavior. In this work, we target the problem of "playing fetch" with a

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

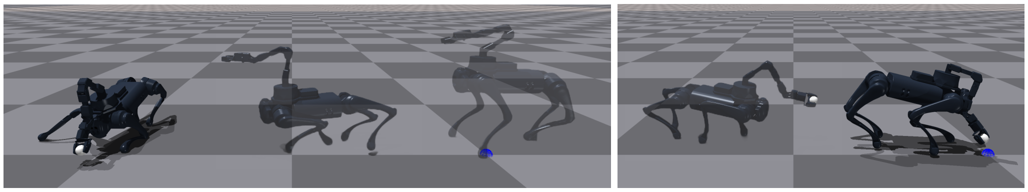

Figure 1: **Example trajectory.** Image sequence from our method of a robot dog running towards a thrown ball, picking it up before the ball comes to rest, and returning to the origin to drop off the ball.

quadruped robot because we believe it exemplifies a hard control task involving both locomotion and manipulation of dynamic objects.

We propose a hierarchical system with a task-independent low-level policy that learns to convert directional commands for the base and position offsets for the gripper to joint torques, and a high-level policy that translates object goals to commands for the low-level policy, balances speed and stability and learns task-specific knowledge.

Figure 1 shows an example of our method in action. Figure 2 shows the schematics of the proposed architecture, with the high-level and low-level control policy.

Our contributions are as follows. **(a)** A Hierarchical Reinforcement Learning (HRL) framework for jointly training a movable base and a robot arm to solve arbitrary control tasks. **(b)** Experiments demonstrating that the method can be applied to solve the problem of playing fetch with a quadrupedal robot. **(c)** Comparisons illustrating how our high-level policy outperforms a PD controller.

All experiments were conducted on the Unitree Go1 quadruped with the Interbotix ReactorX 150 arm in simulation but with the manufacturer's URDF robot models and realistic (manufacturer) dynamics. If this works is accepted, we will present a demo on a real robot at the workshop. Videos of the robot playing fetch can be found at our project website:
`https://robotdogfetch.github.io`

2 Related Work

Control for Legged Mobile Manipulators. Broadly speaking, control methods for mobile manipulators can be placed in two categories: Model Predictive Control (MPC) and learning-based control. MPC can produce a simple control policy for complex systems and provides generic consideration of constraints and complex control goals [6, 7]. However, MPC requires a faithful dynamical model based on forces and torques and can suffer from long re-planning times in dynamic scenes.

In this work, our goal is to provide a policy that can solve legged manipulation tasks with increased robustness and involving dynamic objects. However, manipulation so far has required more precise control, combining MPC and RL methods [8]. By contrast, our method does not require an exact specification of the dynamics of the robot to work. Only very recently, Fu et al. [9] showed that a joint policy can be learned for the base and arm. This work, however, leaves adding task-specific knowledge as an exercise to the reader. In our work, we address this explicitly through the high-level policy that gives the system the autonomy to solve a task without remote control.

Hierarchical Reinforcement Learning. Hierarchical Reinforcement Learning frameworks [10, 11, 12] typically extend the usual notion of action to include options, a closed-loop policy for taking a sequence of actions over a period of time. The concept of an option is quite broad and includes turning around, picking up an object, and driving a commute, as well as lower-level actions such as joint torques. Many such methods train both low and high level policies jointly [13, 14, 15]. The high-level policy operates at a slower frequency than low-level policy options, which may have a fixed or learned duration. For example, [16] used a gradient-based approach to model the termination

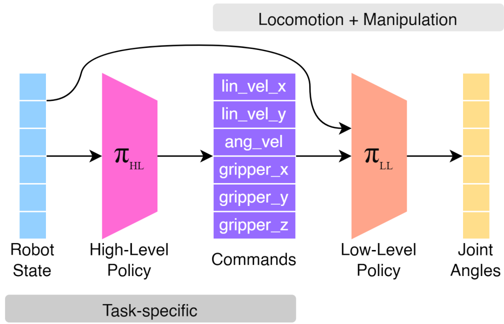

Figure 2: **Abstract policy network**. The high-level policy module takes as input a state vector that describes the robot and world state and produces a command vector. This command vector is then fed into the low-level policy to produce the robot's actuation. The high-level policy and low-level operate at different frequencies, with the low-level policy having to produce joint angles for multiple timesteps to achieve a goal specified by the command vector.

of semi-Markov options by a logistic distribution on a cumulative measure of the features observed during the execution of that option. In this work, low-level policy options correspond to motor actuations which can benefit from very high-frequency control to manage high aleatoric uncertainty in real-world environments, and as a result we fix the duration of low-level policy options.

3 Method

Our method uses a 2-layer hierarchical policy (see Fig.2), where the low-level policy learns locomotion and relative end effector control with respect to the base (based on external commands), and the high-level policy learns to generate commands for the low-level policy for a given task.

Background. We use Nvidia's Isaac Gym simulator to train thousands of quadrupeds in parallel on a GPU to follow commands [17]. In their work, the agent and in our work, the low-level policy is trained via Proximal Policy Optimization (PPO) [18] to obey linear and angular velocity commands. Each command is a 3-dimensional vector, representing the desired linear velocities along the x and y axes (both of which are parallel to the ground) and the yaw rate (rotation speed around the "up" axis). At each timestep, the policy outputs a 12-dimensional vector corresponding to joint angles for the joints of the quadruped (4x hip, upper leg, lower leg). Similar to Rudin et al. [19], we use a weighted combination of 10 reward terms to teach the robot dog to walk in a physically plausible way, e.g., by rewarding the feet to not drag along the floor, or to exert low torques. The most important reward terms are $r_{tracking\_lin\_vel}$, which incentivizes the agent to match the robot body's linear velocity to the first two command terms and $r_{tracking\_ang\_vel}$, which does the same for the yaw rate.

Task & Robot. Our experiments are performed on a simulated Unitree Go1 robot with a manipulator attached to the top of the robot body (see Fig.1). The mounting location was chosen to optimize balance while maximizing forward reach. We use the Interbotix ReactorX 150 robot arm due to its light weight. To train the arm as part of the low-level policy, we added to the observation space the positions and velocities of all joints, to the action space the 5 degrees of freedom, and to the command vector 3 elements corresponding to the desired position of the end effector with respect to the base. For our fetch task, we use a rubber ball and a simple grasping model that closes the gripper when the ball is in the center of the gripper.

High-level Policy. Our high-level policy also uses a 3-layer MLP architecture for both actor and critic networks with $[256, 256, 256]$ hidden units and ELU non-linearities in between. The observations are task-specific and consist of linear/angular velocity of the base, projected gravity, distance between end effector and goal, the current relative position of the end effector, the direction of the goal, and the current heading of the base. The action space consists of the 6 commands comprising the goal conditioning of the low-level policy.

Reward Terms. The high-level policy follows the same PPO training procedure as the low-level policy but with different reward terms. We have reused the existing reward terms and coefficients for action rate (discouraging large changes between timesteps), and large action values (discouraging

	Success (%)	Ball Pickup (%)	Ball Pickup Time (s)	Goal Time (s)
PD Controller & π_{LL}	0 %	3 %	2.7±0.4	n/a
No Random Command	8 %	18 %	1.5±1.1	0.8±0.2
Ours	**97 %**	**99** %	**1.6**±1.6	**1.0**±0.7

Table 1: **Results**. We show the results of our method compared to an ablation ("no command frequency randomization") and a PD controller. We list the success rate, which includes ball pickup and return within 10s, ball pickup rate, average time to pick up the ball, normalized by distance of pickup, and average time to return the ball to the goal (also normalized by ball distance).

extreme joint positions), and we added the terms $r_{goal\_distance}$ (Euclidean distance between gripper and goal), $r_{goal\_distance\_eps}$ (bounded exponential reward for minimizing the distance between gripper and goal), and r_{facing_goal} (encouraging facing towards the goal) - implementation details can be found in the appendix (App.A.1). We also note that the last reward term is both cosmetic and practical for this task as facing the ball reduces the risk of accidentally kicking it further away.

4 Experiments

Experimental Setup. We train our low-level policy for 1,500 timesteps with 4096 parallel environments with a fixed command sampling rate of $\frac{1}{10}$ Hz. We fine-tune the same low-level policy for another 1,500 steps with a variable command sampling rate uniformly sampled from $[\frac{1}{4}, \frac{1}{16}]$ Hz. We found that this domain randomization leads to better generalization. We compare the fine-tuned policy to one that was only trained on the default command resampling time (marked as "No Random Command"). The high-level policy was trained for 3,000 timesteps, corresponding to approximately 2 hours of wall clock time on a single Nvidia Geforce RTX 2080 Ti GPU, which makes this approach suitable to be retrained to a wide range of tasks. To illustrate the necessity of the high-level policy in communicating the task goal to the low-level policy, we also compare to the setting where we replace the high-level policy with the PD controller. The PD controller was manually tuned to strike a balance between stability and speed and to face the goal in order to prevent accidental kicks. The PD controller minimizes the distance between the end effector and the target ball. The target ball is thrown from a position close to the robot in a random direction and the robot has to retrieve the ball and return it to the origin point at $(0, 0, 0)$ (marked in blue in Fig.1) within 10s. We repeat this experiment with 100 random throws for each approach.

Results. The results are listed in Tab.1. We can see that our method solves the task with high efficiency. The few failure cases were due to sharp turns when the robot was moving at a high speed. The results also highlight the necessity of randomizing the control frequency during training, since the ball is a moving target and commands change very quickly, especially in the beginning of an episode. When investigating the runs that use a PD controller, we noticed an interesting detail: The PD controller is often able to approach the ball and lower the gripper to a position right above but it is not able to bend the body of the robot towards the goal. And indeed, there is no way in which the high-level policy can communicate this directly to the low-level policy. The goal command for the gripper is relative to the body of the robot, i.e., moves with the base. In other words, the high-level policy found a way to "trick" the low-level policy into lowering its body when approaching the goal.

Conclusion. We presented a hierarchical control approach for combining quadruped locomotion and robot arm manipulation and we demonstrated its efficacy on a difficult rolling ball pickup task. There are still several avenues for improvement of this method that we would like to explore. So far, we have only trained the method on flat ground but going forward, we want to investigate how the method handles bumpy terrain and stairs. We are also excited to train the high-level policy on different tasks with the same low-level policy to analyze how well our method generalizes to new problems and environments.

References

[1] F. Rubio, F. Valero, and C. Llopis-Albert. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2):1729881419839596, 2019.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[3] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

[4] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[5] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019.

[6] S. Zimmermann, R. Poranne, and S. Coros. Go fetch! - dynamic grasps using boston dynamics spot with external robotic arm. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4488–4494, 2021. doi:10.1109/ICRA48506.2021.9561835.

[7] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter. A unified mpc framework for whole-body dynamic locomotion and manipulation. *IEEE Robotics and Automation Letters*, 6 (3):4688–4695, 2021.

[8] Y. Ma, F. Farshidian, T. Miki, J. Lee, and M. Hutter. Combining learning-based locomotion policy with model-based manipulation for legged mobile manipulators. *CoRR*, abs/2201.03871, 2022. URL https://arxiv.org/abs/2201.03871.

[9] Z. Fu, X. Cheng, and D. Pathak. Deep whole-body control: Learning a unified policy for manipulation and locomotion. *arXiv preprint arXiv:2210.10044*, 2022.

[10] R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. In *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10*, NIPS '97, page 1043–1049, Cambridge, MA, USA, 1998. MIT Press. ISBN 0262100762.

[11] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *CoRR*, cs.LG/9905014, 1999. URL https://arxiv.org/abs/cs/9905014.

[12] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999. ISSN 0004-3702. doi:https://doi.org/10.1016/S0004-3702(99)00052-1. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

[13] P. Bacon, J. Harb, and D. Precup. The option-critic architecture. *CoRR*, abs/1609.05140, 2016. URL http://arxiv.org/abs/1609.05140.

[14] S. Yu, S. Rammohan, K. Zheng, and G. Konidaris. Hierarchical reinforcement learning of locomotion policies in response to approaching objects: A preliminary study, 2022. URL https://arxiv.org/abs/2203.10616.

[15] C. Li, F. Xia, R. Martin-Martin, and S. Savarese. Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators, 2019. URL https://arxiv.org/abs/1910.11432.

[16] G. Comanici and D. Precup. Optimal policy switching algorithms for reinforcement learning. volume 2, pages 709–714, 01 2010. doi:10.1145/1838206.1838300.

[17] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox. Gpu-accelerated robotic simulation for distributed reinforcement learning. In *Conference on Robot Learning*, pages 270–282. PMLR, 2018.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[19] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. *CoRR*, abs/2109.11978, 2021. URL https://arxiv.org/abs/2109.11978.

A Appendix

A.1 Implementation Details

The reward terms for the high-level policy are as follows:

$r_{goal\_distance}$ (normalized Euclidean distance between gripper and goal) is calculated as

$$r_{goal\_distance} = \frac{d_{MAX} - d_{curr}}{d_{MAX}},$$

where

d_{MAX}.. maximal distance a goal can spawn from the robot's base.
d_{curr}.. current distance between end effector and goal.

$r_{goal\_distance\_eps}$ (bounded exponential reward for minimizing the distance between gripper and goal) is calculated as

$$r_{goal\_distance\_eps} = e^{\frac{-d_{curr}}{\sigma}},$$

where

σ.. hyperparameter, set to 0.1 in our experiments.

r_{facing_goal} (encouraging facing towards the goal) is calculated as

$$r_{facing\_goal} = ||\vec{fwd} - \vec{goal}||,$$

where

$\vec{fwd}$.. unit vector representing the forward direction of the robot body, parallel to the ground plane.
$\vec{goal}$.. unit vector representing the direction towards the goal, also ground-parallel.

A.2 Hyperparameters

Parameter	Value
σ	0.1
coefficient for $r_{goal\_distance}$	1.0
coefficient for $r_{goal\_distance\_eps}$	1.5
coefficient for r_{facing_goal}	0.5