QUADRUPED ROBOTS WALK BIPEDALLY USING REWARD-BASED METHODS

Liangsheng Yin
Reinforcement Learning Course, ACM 2021
Shanghai Jiao Tong University
liangsheng.yin@outlook.com

ABSTRACT

Most quadruped robots are trained for a quadrupedal gait using reinforcement learning techniques such as Rapid Motor Adaption (RMA). However, training them for **bipedal walking within the same framework** holds great potential. This course project introduces a reward-based method to train quadruped robots for bipedal walking on *plain* and *stairs* terrains. It achieves favorable results in metrics such as linear tracking velocity, total climbing height, and fall prevention. Additionally, it demonstrates that RMA is a feasible method for training quadruped robots for bipedal walking in a simulated environment, showcasing its significant robustness and adaptability.

1 INTRODUCTION

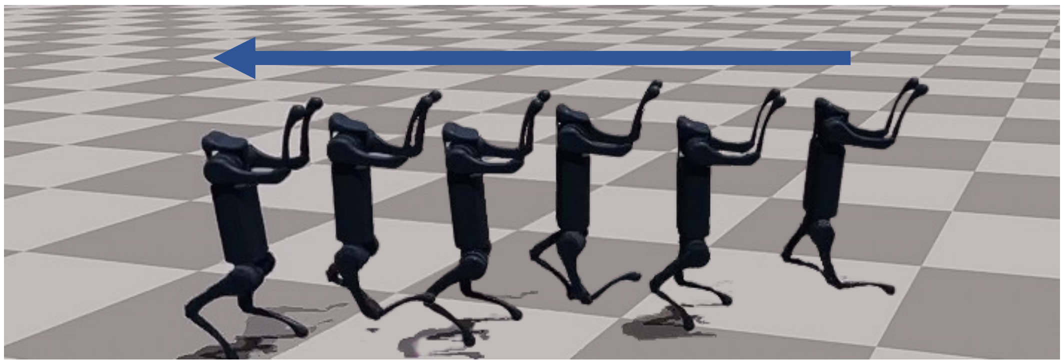

Figure 1: Bipedal Walking Posture on Plane Terrain

Training quadruped robots to walk on four legs is a well-established field in robotics. This course project is based on the famous Rapid Motor Adaptation (RMA) algorithm and its open-source implementation (Kumar et al., 2021). Since the introduction of RMA, other research aimed at enhancing this algorithm has been proposed, including efforts for more precise and safe control (Saviolo & Loianno, 2023), as well as integration with egocentric vision Agarwal et al. (2023).

Bipedal robots represent another important research topic, as they often resemble humans and possess significant potential for deployment in real-world industries. Various studies have been proposed for human-like robots with whole-body control (Cheng et al., 2024). However, these robots often have too many degrees of freedom, making it challenging to define the reward functions necessary for control.

These two topics bring an interesting idea: Can we make the quadruped robots behave like straight-standing dogs, walking bipedally while maintaining balance? Our framework, the RMA algorithm, is known for its rapid adaptation capabilities. However, the teacher training phase also embraces a classic reinforcement learning process with the PPO algorithm (Schulman et al., 2017), which means we can change the reward functions to make the teacher walk bipedally and then distill the ability to the student. The Unitree A1 quadruped robot possesses only 12 degrees of freedom, fewer than a human-like robot, making this goal more achievable. Recent research (Li et al., 2024) has also shown a similar result to let quadruped robots stand bipedally and imitate human action.

Our method includes adjusting the robot's initial rotation and joint angles, modifying the commands, and introducing a set of reward functions designed for bipedal walking on plain and stairs terrains respectively. Our result shows great balancing ability even when climbing steep stairs. The Figure 1 shows the bipedal walking posture on the plane terrain with a time interval of 0.1s and the blue arrow represents the moving direction. The Figure 2 shows the process of climbing out of a square pit with a time interval of 0.5s and the robot keeps a bipedal orientation while climbing the stairs.

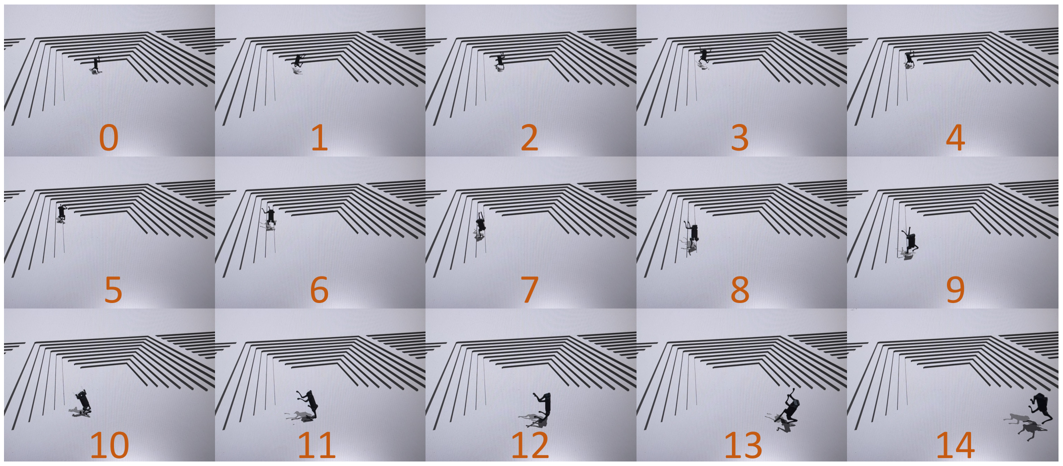

Figure 2: Process of Climbing out of a Square Pit

2 Reward-Based Method to Bipedal Walking

2.1 Initial Rotation and Joint Angles

The first challenge is how to make the initially horizontal robot's body (we call it base in the code) stand up since the default body's position and rotation are designed for walking with four legs. When trying to raise the robot's base from the ground to a standing position, it requires an extra training phase to let the network learn how to stand up slowly and robustly. However, when later training from the standing position to walking, the network may be disturbed by the change of rewards and lose the ability to stand up. To solve this problem, we adjust the initial status of the robot, making it ready for bipedal walking at the beginning of the training process. The parameters are shown in Table 1 and Table 2.

Dimensions	Default Parameter (quat)	Bipedal Walking Parameter (quat)
x	0.0	0.0
y	0.0	-1.0
z	0.0	0.0
w	1.0	1.0

Table 1: Rotation Parameters

2.2 Commands Modification

The original training framework assigns an x-axis velocity and a random angular velocity for the xy plane, as shown in Figure 3. When we make the base stand up straight, we must remove all velocity towards the x-axis, as it now represents the vertical direction. The original x-axis velocity is for the

Joint Name	Default Parameter (rad)	Bipedal Walking Parameter (rad)
FL Hip	0.1	0.2
RL Hip	0.1	0.7
FR Hip	-0.1	0.2
RR Hip	-0.1	0.7
FL Thigh	0.8	-1.0
RL Thigh	1.0	1.5
FR Thigh	0.8	-1.0
RR Thigh	1.0	1.5
FL Calf	-1.5	-2.2
RL Calf	-1.5	-0.8
FR Calf	-1.5	-2.2
RR Calf	-1.5	-0.8

Table 2: Joint Parameters

`tracking_lin_vel` reward. Therefore, if we want to train the robot to walk bipedally, we should add a `z`-axis velocity and a reward for `tracking_lin_vel_z`. Alternatively, we can use another reward, `lin_vel_z`, instead (see subsection 2.3). This approach avoids modifying too much code in the training framework.

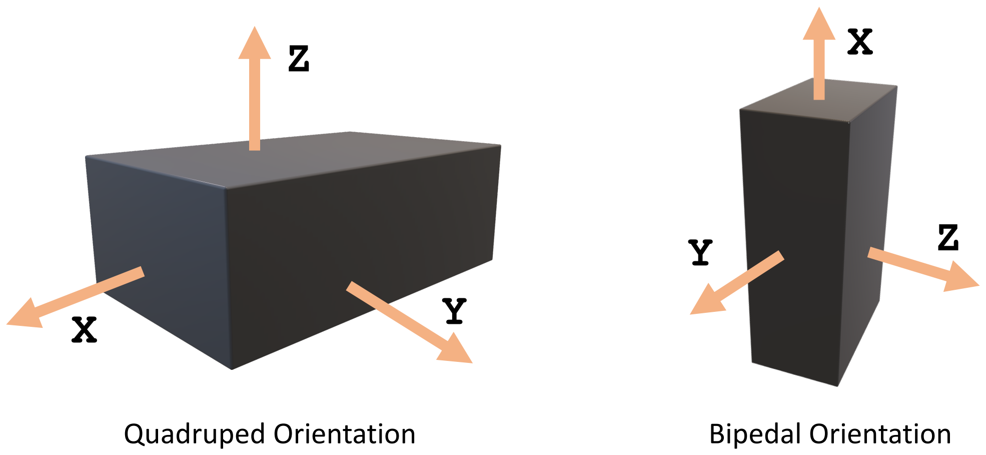

Figure 3: Base Orientation of Quadruped and Bipedal Walking

2.3 Rewards for Plane Walking

2.3.1 Reward Linear Velocity in Z Axis.

As demonstrated in the bipedal example of Figure 3, to encourage forward movement in bipedal walking without directly adding a `z`-axis velocity to track such direction, it is pivotal to motivate the robot towards forward motion by monitoring the `z`-axis velocity. To this end, we introduce a reward function, `reward_lin_vel_z`, as detailed in 1. However, during bipedal locomotion, *if the robot falls, the* `z`*-axis velocity may surge to excessively high levels, creating the risk of a gradient explosion within the network. To prevent this, we cap the velocity at a maximum value, which is set to 9.0 in our experiments.*

Algorithm 1 Calculate Reward Based on Linear Velocity in Z Axis

```
function RewardLinVelZ
    Parameters: max_z_vel ← $9.0$
    Initialize: lin_vel_z ← Square(base_lin_vel_z)
    Return Clip(lin_vel_z, $0.0$, max_z_vel)
end function
```

2.3.2 Penalize Other Directions' Linear and Angular Velocity.

To incentivize forward movement, it's also important to penalize linear and angular velocities in other directions. For instance, a reward function `penalize_ang_vel_z`, as shown in 2, is added to penalize the angular velocity in the `z`-axis, corresponding to the robot's rotation around the `z`-axis. The penalty is calculated using the exponential function of the negative square of the angular velocity, a standard method in reinforcement learning to discourage high values of a variable.

Algorithm 2 Calculate Panelty Based on Angular Velocity in Z Axis

```
function PenalizeAngVelZ
    ang_vel_z ← Square(base_ang_vel_z)
    return exp(-ang_vel_z/$0.25$)
end function
```

2.3.3 Reward Bipedal Orientation.

To promote continuous upright posture in the robot, we introduce a crucial reward function, `reward_bipedal_orientation`, as detailed in 3. This function penalizes the robot's deviation from the vertical axis. The reward is computed as the sum of the squares of the projected gravity vector on the `yz` plane, which is the plane on which the robot stands. This reward is structured to be positive, with higher values indicating a more vertical orientation of the robot.

Algorithm 3 Reward Bipedal Orientation

```
function RewardBipedalOrientation
    return $\sum$(Square(projected_gravity_yz))                ▷ Sum over dim=1
end function
```

2.4 Rewards for Stair Climbing

The reward functions previously described are optimized for bipedal walking on level ground. *However, applying the same reward structures to stair climbing scenarios within a square pit may result in suboptimal performance, where the robot fails to effectively ascend the stairs. Instead, it may persist in moving in a circular path at the pit's base.* To address this issue, we introduce the `total_climb_height` 4 reward, designed to motivate the robot to tackle the stairs successfully. This reward is computed based on the average of the total climb height, which equals the sum of heights that the robot's root state has ascended, each multiplied by the height interval which denotes the elevation difference between two successive levels of terrain. In our experiments, we set the minimum height threshold at -0.5.

Algorithm 4 Reward Total Climb Height

```
function RewardTotalClimbHeight
    climb_height ← root_states_h $+$ terrain_levels $\times$ height_interval
    return Mean((climb_height $-$ MIN_H) $\times$ terrain_levels)
end function
```

3 Experiment Setup

3.1 Reward Scales

The detailed reward functions used in the experiments are as follows:

- Torques: Penalize high joint torques.
- Action Rate: Penalize high action rate.
- z Linear Velocity: Encourage forward movement.
- x Linear Velocity: Penalize movement in the x direction.
- y Linear Velocity: Penalize movement in the y direction.
- x Angular Velocity: Penalize rotation around the x axis.
- z Angular Velocity: Penalize rotation around the z axis.
- Bipedal Orientation: Encourage vertical orientation.
- Bipedal Fall Down: Penalize falling down.
- Total Climb Height: Encourage climbing stairs.
- Collision: Penalize collision.

The reward scales for the plain and stairs terrains are shown in Table 3.

Reward Name	Plain Terrain	Stairs Terrain
torques	-0.0002	-0.0002
action rate	-0.05	-0.05
z linear velocity	1.0	1.5
x linear velocity	0.5	2.0
y linear velocity	1.0	1.0
x angular velocity	5.0	5.0
z angular velocity	1.0	1.0
bipedal orientation	-5.0	-5.0
bipedal fall down	-10.0	-10.0
total climb height	0	10.0
collision	-1.0	0

Table 3: Reward Scales

4 Result and Discussion

4.1 Plane Walking

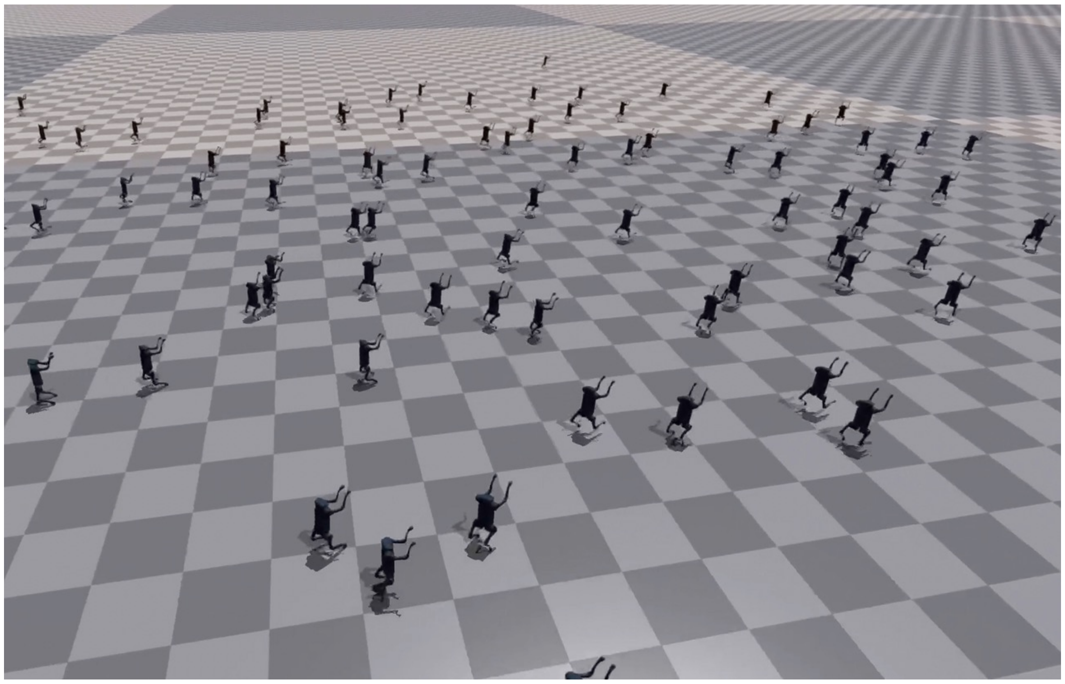

Figure 4: Bipedal Walking on Plane Terrain

The teacher and student models exhibit comparable performance on flat terrain, as demonstrated in the experiments shown in Figure 4, where 128 quadruped robots are simulated to walk bipedally across a vast planar surface. Results indicate that the average z-axis velocity can reach up to $3m/s$, with a fall probability within 2%.

4.2 Stairs Climbing

When it comes to climbing stairs, the teacher model slightly outperforms the student model in terms of balance. With staircases comprising 10 levels of `trimesh` stairs, each level separated by approximately 0.15m, the teacher model successfully ascends to the 4th level, whereas the student model peaks at the 3rd level. Across all 10 `trimesh` stair levels, the teacher model exhibits a lower fall probability, in comparison to the student model's higher fall likelihood of about 10%.

4.3 Stairs Climbing

References

Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 403–415. PMLR, 14–18 Dec 2023. URL `https://proceedings.mlr.press/v205/agarwal23a.html`.

Xuxin Cheng, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang. Expressive whole-body control for humanoid robots, 2024.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots, 2021.

Yunfei Li, Jinhan Li, Wei Fu, and Yi Wu. Learning agile bipedal motions on a quadrupedal robot, 2024.

Alessandro Saviolo and Giuseppe Loianno. Learning quadrotor dynamics for precise, safe, and agile flight control. *Annual Reviews in Control*, 55:45–60, 2023. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2023.03.009. URL `https://www.sciencedirect.com/science/article/pii/S1367578823000135`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.