Flexible Framework for Agile Dribbling with Cyberdog2

Haoyang Weng, Chuan Liu, Xiang Ji, Xuan Qi
Institute for Interdisciplinary Information Sciences
Tsinghua University, Beijing
{wenghy22, liuchuan22, jix22, qi-x22}@mails.tsinghua.edu.cn

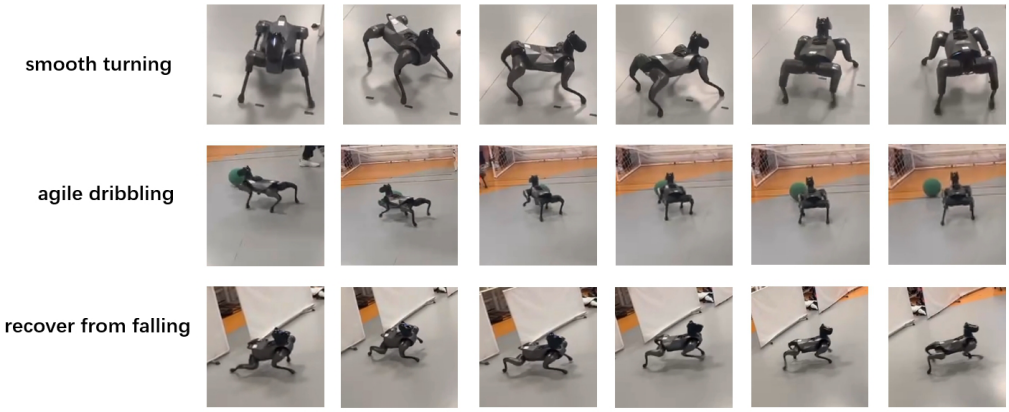


Figure 1: Demonstration of Various Agile skills: (top row) smooth turning maneuvers, displaying precise rotational control; (middle row) agile dribbling, exhibiting dynamic movement and object interaction; (bottom row) recovery from falling, demonstrating policy robustness.

Abstract

This report presents a flexible framework for agile dribbling with the Cyberdog2 quadruped robot. We utilized reinforcement learning techniques to train effective control policies for locomotion and dribbling tasks. Our approach addresses key challenges in sim-to-real transfer, including inference frequency optimization, default pose adjustment, and PD control parameter tuning. The resulting system demonstrates robust performance in real-world soccer playing scenarios.

1 Introduction

Recent advancements in robotics have shown the potential of deep reinforcement learning for addressing complex tasks involving quadruped robots, such as locomotion and ball manipulation. Our work focuses on developing an integrated system for soccer-playing skills using the Cyberdog2 platform. We propose a flexible framework that combines learned policies for locomotion and dribbling with a rule-based high-level planner.

Our approach builds upon previous work in quadruped locomotion and dribbling, adapting these techniques to the Cyberdog2 platform. We address several challenges in sim-to-real transfer, including inference speed optimization, pose adjustments, and control parameter tuning. The resulting system demonstrates effective performance in real-world scenarios, showcasing the potential of learning-based approaches for complex robotic tasks.

Preprint. Under review.

2 Related Work

Recent advancements in robotics have demonstrated deep reinforcement learning as a promising approach for addressing tasks that require complex dynamics and contact-rich manipulation, particularly in the context of quadruped robots performing locomotion, dribbling, and kicking.

[4] addressed the problem of under-specified rewards by learning a policy that can be used to generate a wide variety of behaviors given specified command inputs with task-auxiliary rewards. Their policy enables the robot to generalize to different out-of-distribution environments, assisted with a human teleoperator issuing commands in real-time.

[3] considers the task of dribbling a soccer ball in the wild, which requires task-oriented coordination of all the robot's joints. To tackle the Sim2Real gap, they finetuned a YOLO model for deployment and added vision noise domain randomization. They also integrated a recovery policy that enables the robot to stand up after a fall. Their system can be adapted to complex ball-terrain dynamics and maintains precision even when the robot is close to the ball.

[1] introduces a novel feedback control loop to add supervision to the training process, to overcome the limitations of DribbleBot on smooth surfaces. They achieved this by using a designed heuristic to encourage the robot to overspeed to stop the ball with dynamic supervision in the form of shaping rewards.

Apart from dynamic movements, [2] explores using one of the front legs of a quadruped robot to perform kicks when the other three legs are kept stable. They leveraged hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot, employing Bezier curves as the interface between a low-level controller and a high-level planner, both trained using deep reinforcement learning.

3 Method

Our method comprises three main components: policy training, deployment framework, and sim-to-real transfer optimizations. Figure 2 illustrates the overall pipeline of our approach.

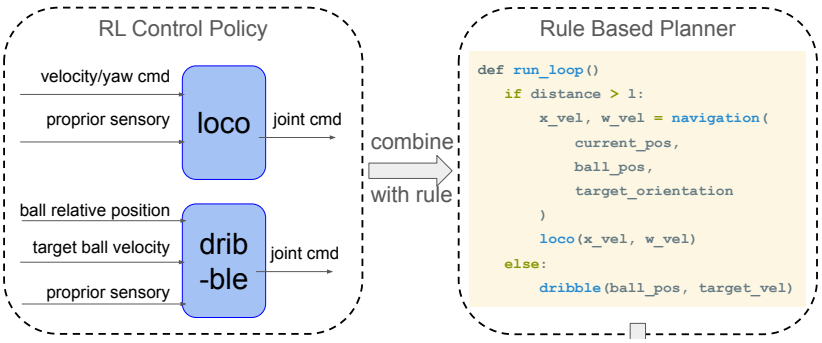


Figure 2: Pipeline of training and deployment

3.1 Policy Training

We utilized the DribbleBot and Walk These Ways frameworks to train reinforcement learning policies for dribbling and locomotion tasks.

Observation: Policy inputs were derived from both proprioception (motor feedback) and the relative ball position required by the dribble policy.

Actions: The output of the policy is joint target positions, which were executed via the motor SDK using PD control.

Training: We directly used the open-sourced DribbleBot and Walk These Ways repositories to train the RL policies. Our pipeline was interleaved between training and deployment, allowing us to add new regularization rewards, identify potential crucial environment parameters, and fine-tune the reward scales.

3.2 Deployment Framework

We developed a flexible deployment framework that includes:

- A Python binding of the **low-level motor SDK** to ease further deployment and testing of rule-based control strategies.
- A **location server** utilizing the upstream machine to provide global positions of the robot and ball, converting them to ball positions relative to the robot.
- A **rule-based planner** that switches between the locomotion policy and the dribble policy based on the game status and relative distance between the robot and the ball.

Figure 3 illustrates the deployment framework we used to deploy the trained policy. Except for the low-level motor SDK, all of the code in the framework are implemented by ourselves and open sourced on GitHub.

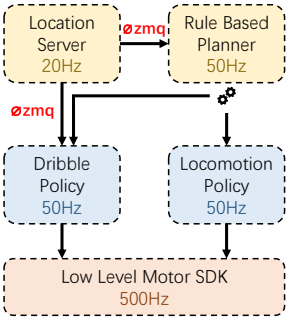


Figure 3: Deployment Framework

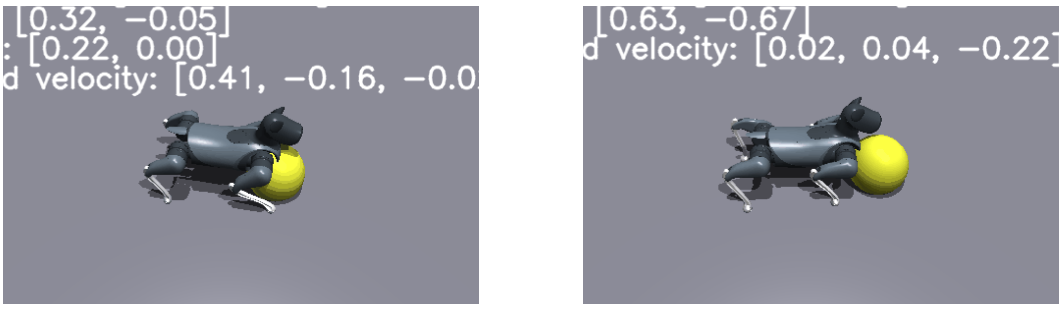
3.3 Sim-to-Real Transfer Optimizations

In this section, we discuss the challenges encountered and the corresponding **measures** taken to successfully deploy DribbleBot on the Cyberdog2, aiming to provide insights for others pursuing similar real-world deployments and demonstrate our engineering efforts.

3.3.1 Dribble

Inference Frequency Optimization. Initially, the motion of the robot appeared laggy due to an unsatisfactory inference frequency. With full precision (fp32) inference on the Jetson Xavier NX onboard CPU, the control loop ran at 20Hz. However, by leveraging **half-precision (fp16) inference**, we achieved a maximum frequency of 180Hz, satisfying our required inference frequency of 50Hz.

Default Pose Adjustment. The action out by the policy is interpreted as an offset added to the default initial position. With the default standing pose, the robot would climb over the ball during dribbling, preventing its front feet from contacting the ground, and will stuck on this position. To address this issue, we **modified the default joint positions to lower the robot's body**, as illustrated in 4.



(a) High default pose

(b) Low default pose

Figure 4: Comparison of Default position

PD Control Parameter Tuning. We encountered a significant sim-to-real gap in the PD control parameters (Kp and Kd). In the simulation, using Kp=20 and Kd=0.5 led to good results, but led to soft leg movements when deploying in real world, causing the robot to be unable to stand still and falling forwards. Increasing Kp to 40 60 enabled a stable stance pose in the real world. However, increasing Kp and Kd in the simulation resulted in training instability, with the policy either becoming overly conservative (standing still) or too aggressive (performing large actions leading to irregular poses on the ground), indicating the need for reward tuning. Due to these issues, we fixed Kp and Kd in simulation to 20 and 0.5, respectively. For real-world deployment, we **manually searched**

for the optimal set of control parameters, ultimately using Kp=60 and Kd=2 to achieve stable performance.

Location Server Delay Modelling. In real world deployment, we found the frequency of the upstream machine is about 5Hz, which means a delay of 10 steps relative to policy inference step. To mitigate this issue, we **modeled a detection decimation of 10 in the simulation**, resulting in improved performance.

Mitigating Detection Bias and Noise. We manually calibrated the detection location by place the ball on a 1x1m grid and record the detection location, then **fit a homography transformation to correct the detection location**. As shown in Figure 5, the detection location is corrected to the center of the ball. Due to varying lighting conditions and environmental factors, the upstream machine occasionally provided inaccurate ball location estimates. To counteract this detection noise, we implemented a **history averaging filter** that smoothed the ball position over a window of 20 steps. This filtering technique helped stabilize the ball position input, mitigating the impact of transient noise and enabling more reliable dribbling performance.

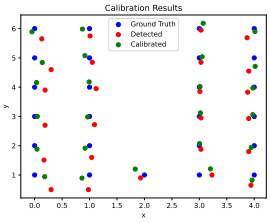


Figure 5: Calibration of detection location

3.3.2 Locomotion

Locomotion mostly adapts the framework of Dribble to ensure consistency after Sim-to-Real transfer.

Locomotion Gait Simplification. Initially, we attempted to train a policy following the original setting of `Walk These Ways`, with gaits switching between four styles: pronking, trotting, pacing, and bounding. But we soon found that this approach mismatched with our goal as it took long time to train and is fragile to parameter setting. Our dog differs from the Unitree go1 dog used in the paper mainly in total mass and relative position of base. These slight changes make the dog hard to stand still in default pose and therefore affect the overall quality of locomotion. To simplify the training step and equip our dog with basic ability to walk, we use **single gait training**, by carefully choosing $\theta^{\text{cmd}} = (0.5, 0, 0)$ as the only gait to be sampled during training.

Reward scale adjustment. We carefully chose a set of reward scales to **reduce z direction velocities, encourage command velocity tracking, and avoid y direction rotations**. Below figure illustrates our dog's capability of tracking x direction velocities even if the command velocity is high (e.g., $1.0m/s$) while maintaining reasonable joint positions.

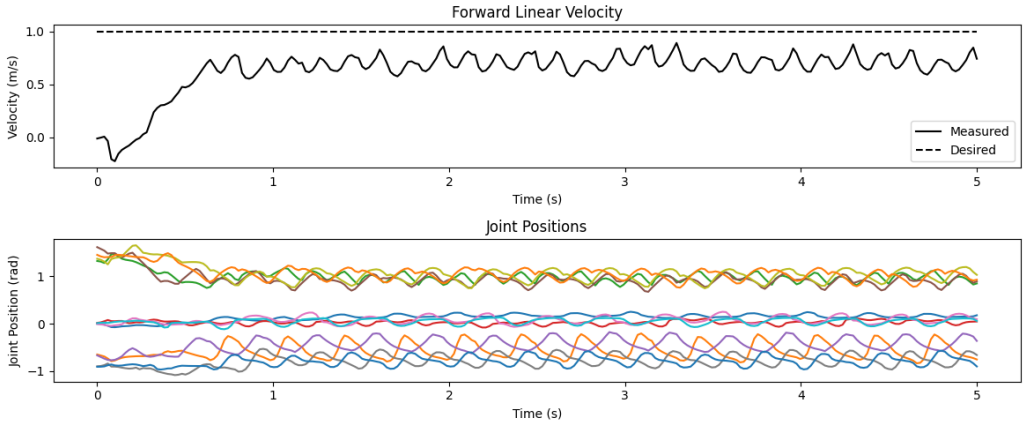


Figure 6: Improved linear velocity tracking

4 Contributions

1. Haoyang Weng: Dribble policy training, deployment framework setup, and writing.

2. Chuan Liu: Locomotion policy training and deployment, and writing.
3. Xiang Ji: Rule based planner design and deployment, making diagrams and writing.
4. Xuan Qi: Locomotion policy training, assisting rule based planner deployment, making diagrams and writing.

5 Discussion and Future Work

The flexible framework developed in this project showcases the potential of learning-based approaches for complex quadruped robot tasks. By addressing key challenges in sim-to-real transfer, we have demonstrated the feasibility of deploying reinforcement learning policies on real-world robotic platforms.

Future work could explore:

- Further optimization of the high-level planner for more complex game strategies.
- Integration of imitation learning or teacher-student distillation to merge sub-policies into a unified neural network policy.
- Online adaptation and reinforcement fine-tuning in real-world environments to further improve performance.

References

[1] Yutong Hu, Kehan Wen, and Fisher Yu. Dexdribbler: Learning dexterous soccer manipulation via dynamic supervision, 2024.

[2] Yandong Ji, Zhongyu Li, Yinan Sun, Xue Bin Peng, Sergey Levine, Glen Berseth, and Koushil Sreenath. Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot, 2022.

[3] Yandong Ji, Gabriel B. Margolis, and Pulkit Agrawal. Dribblebot: Dynamic legged manipulation in the wild, 2023.

[4] Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior, 2022.