Playful DoggyBot: Learning Agile and Precise Quadrupedal Locomotion

Xin Duan[12] Ziwen Zhuang[13] Hang Zhao[13] Sören Schwertfeger[2]

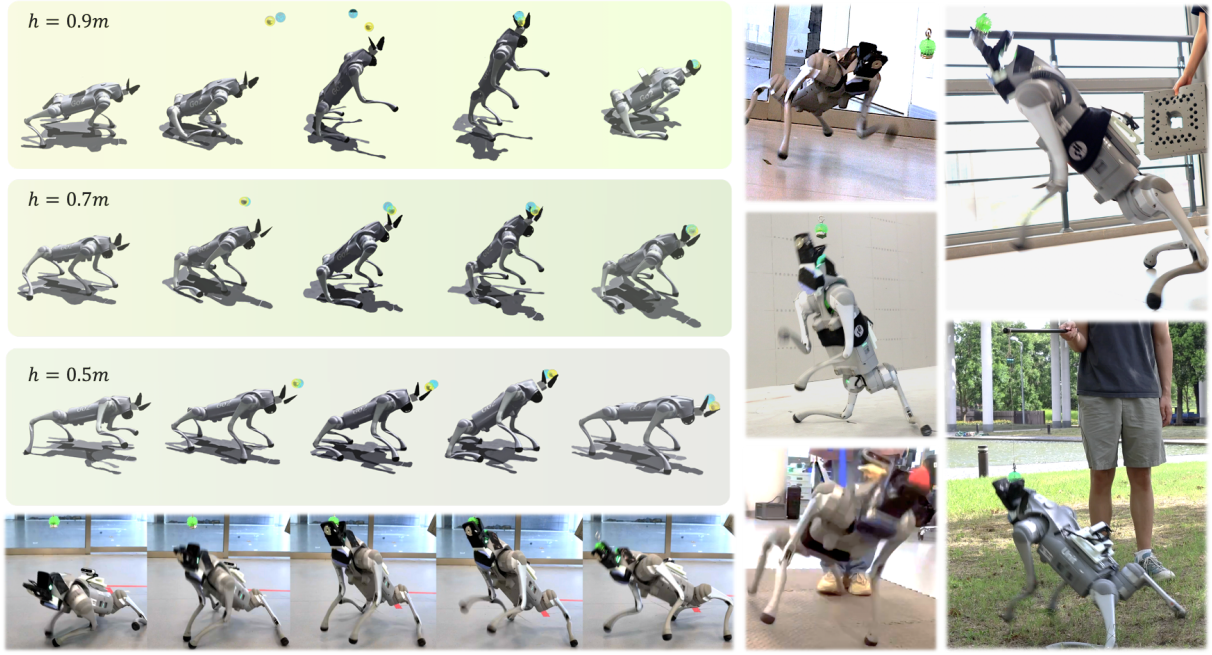

Fig. 1: **Playful DoggyBot.** We present a system to explore the agile and precise movements of quadrupedal robots. A robot dog mounted with a mouth-like gripper can finish the challenging task of leaping up to catch a small target object. Code and videos are available on the Project Webpage: https://playful-doggybot.github.io/.

***Abstract*— Quadrupedal animals can perform agile and playful tasks while interacting with real-world objects. For instance, a trained dog can track and catch a flying frisbee before it touches the ground, while a cat left alone at home may leap to grasp the door handle. Successfully grasping an object during high-dynamic locomotion requires highly precise perception and control. However, due to hardware limitations, agility and precision are usually a trade-off in robotics problems. In this work, we employ a perception-control decoupled system based on Reinforcement Learning (RL), aiming to explore the level of precision a quadrupedal robot can achieve while interacting with objects during high-dynamic locomotion. Our experiments show that our quadrupedal robot, mounted with a passive gripper in front of the robot's chassis, can perform both tracking and catching tasks similar to a real trained dog. The robot can follow a mid-air ball moving at speeds of up to 3m/s and it can leap and successfully catch a small object hanging above it at a height of 1.05m in simulation and 0.8m in the real world.**

I. INTRODUCTION

A common scenario in our daily lives features a trained dog expertly chasing after a fast-moving frisbee and leaping up to catch it just before it hits the ground. The ability of robots to perform agile movements and precise manipulation tasks in dynamic environments is a crucial aspect of their functionality. Successfully catching the objects requires real-time visual perception and the ability to remember and understand the surrounding environment [1]. Visual perception and memory enable the dog to detect and track the trajectory of the target object, and then to accurately judge distance and timing for the leap, knowing the optimal positioning and torque required. However, the ability of robots to act quickly and accurately with real-world objects remains a challenging task that requires exceptional agility and precision. In this work, we equip a quadruped robot known for its agile movements with an end-effector resembling a dog's mouth and train it to track, jump up, and catch a mid-air ball smaller than a tennis ball (with a diameter of no more than 5 centimeters), aiming to explore its agility and precision. The robotic dog demonstrates continuous pursuit capability while the small ball is rapidly and randomly moving suspended from a string and operated by a human.

During high-speed locomotion of legged robots, sensor inaccuracies and latency as well as actuator execution errors become noticeable. In our configuration, the egocentric vision system experiences both rapid horizontal motion and persistent vertical jolts from the legged robot's dynamic gait, which induces pronounced motion blur in captured images, thereby reducing target object detection accuracy. At $1.5m/s$

[1] Shanghai Qi Zhi Institute [2] Key Laboratory of Intelligent Perception and Human-Machine Collaboration – ShanghaiTech University. [3] Tsinghua University

locomotion speed, even a small unexpected delay of $0.05s$ will lead to $0.075m$ positional error, which already exceeds the accuracy requirement in our task. Such conditions require that onboard system be able to adjust its behavior in real-time. In contrast, manipulation tasks demand greater operational accuracy, but they are typically deployed in quasi-static environments where sensor-induced errors are minimal, not requiring real-time computation. Thus, these tasks can utilize complex, large models in the system.

Recent studies tried to resolve this issue, but focused on simpler tasks. In [2], the quadrupedal robot uses its legs to interact with a soccer ball that has a relatively large diameter of approximately $20cm$, which does not require extremely high accuracy. Although [3] employs a tennis ball-sized object, the catching device remains relatively large, thereby allowing greater margin for error. [4] disentangles the agile maneuvering from precise manipulation through two operation stage by moving to the target with agile locomotion and perform the manipulation in a relatively slow manner, so that these two technical requirement do not conflict with each other. Some tasks require both agility and precision [5], [6], but the robot can resolve the trade-off with more action redundancy. For instance, when a quadrupedal robot is trained to climb a certain staircase, it can lower its precision requirements by just lifting its legs higher. The policy learns to act with more torques or take the leap slightly earlier than the exact acting point. In our work, we aim to solve the tracking and catching tasks without much room for action redundancy. Successfully catching demands the small ball to avoid being knocked away by the gripper or other body components of the robot and hit a rectangular catching region that is less than $25cm^2$. The robot cannot improve its hit rate by jumping higher or lower, which would knock the ball away by hitting the edges of the gripper.

The main contributions are as follows:

- We demonstrate a complete system for high-speed and precise quadrupedal manipulation, enabling a real robot to track small objects moving at speeds of up to 3m/s with sudden direction changes, and successfully catch targets at heights of up to 0.8m while in dynamic motion.
- We employ a comprehensive training framework that addresses the agility-precision trade-off through carefully designed reward functions, curriculum learning, and memory-equipped network architectures.
- We validate our simulation-trained policies in simulation and real-world experiments. We analyze key factors affecting catching performance, including the perception noise of the target position, the absolute height of the target and the number of different target heights during training, providing insights into the challenges of dynamic object manipulation for quadrupedal robots.

II. RELATED WORK

A. Legged Agile Locomotion

There is lots of work like the ETH STARLETH robot [7], Boston Atlas robot [8] and the MIT Cheetah robot [9] showing impressive legged locomotion ability including walking on various terrain [10], [11], [12], [13], [14], [15], [16], running [17], jumping [7], stepping over obstacles [18], [9], [19] and even smooth parkour skills [8]. However, these cases using model-based control techniques usually need a lot of engineering efforts for modeling the robots and the environment and suffer from scaling its ability to diverse environments and changes in dynamics. Recently, we have experienced an explosive development of learning-based control techniques that not only accomplish the basic ability we mentioned before [20], [21], [22], [23], [19], [24], but also various fancy locomotion skills including climbing up and down stairs [25], [26], [27], [28], [29], [30], [31], [32], resetting to safe gaits [33], jumping over gaps [31], [5], [6], back-flipping [34], standing up on rear legs [35], [36], moving with damaged parts [37], weaving poles [38] and also parkour skills [5], [6]. There are also combinations of model based control and Deep Reinforcement Learning (DRL), leveraging the advantages of both to improve robustness and generalization [39], [24].

B. Quadrupedal Manipulation

Locomotion focuses more on the robot's own movement capabilities, while manipulation focuses on the robot's ability to interact with objects in the real world. The most direct way to enable quadrupedal robots to do manipulation tasks involves mounting an arm manipulator on it [40], [41], [42], [43], [44], [45]. Since sometimes the four limbs for the quadrupedal robot are redundant for walking, there are also some works using legs as manipulators [46], [47], [48], [49], [2], [50] or with a gripper mounted on the leg [47], [51], [48]. Inspired by the morphology of dogs in nature, we mount a gripper on the head position to enable simple grasp actions.

Previous work has shown the interaction ability, including opening doors [46], [47], [42], [45], pressing buttons [46], [48], [47], and picking and placing objects [48], [42]. [2] works as a soccer goalkeeper to stop a ball flown over using a third-view camera. [3] catches the small flying ball with a net bag using an event camera, requiring the robot to move to the specified location before the key time point. In both of these works, the robot initially stays stationary, waiting until the estimator calculates the ball's landing point before it moves to the designated position at the critical time step. Similar to our approach [4] mounts a 1-DoF gripper to the front of a quadruped, serving as the end-effector. They leverage pre-trained vision-language models (VLMs) to develop a robot system for learning quadrupedal mobile manipulation. Following human commands, the robot performs tasks like climbing onto the bed and then picking up the desired object. In this work, we test the possibility of a robot performing extremely precise action while moving at a high speed with only its onboard sensors and computation. Therefore, our robot dog is equipped with an end effector to run toward the target and capture it.

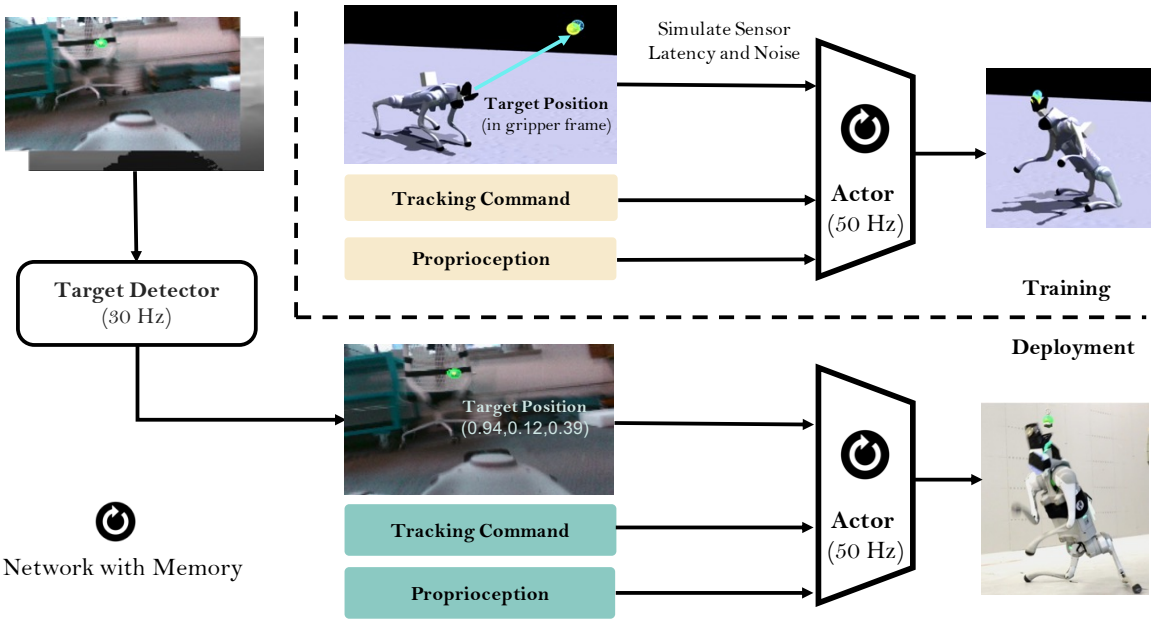

Fig. 2: **System Framework (Pipeline).** We use the policy network trained in the simulation to map the observation input, which includes proprioception and the goal position coordinates computed using the object detector, into goal joint angles. Then the PD controller computes the motor torques with respect to the goal joint angles, current joint angles, and joint velocities, and applies them to the real robot.

III. METHOD

Our method trains a neural network that is mapping the proprioception, the compact target position and the desired robot velocity into joint angle commands. The general pipeline of our approach is shown in Figure 2.

Decoupling the perception system allows each system to operate at its optimal frequency - e.g. our low-level controller runs at 50Hz while the perception pipeline can run at a lower frequency according to the hardware capability and the computation cost. This also allows us to model and handle perception uncertainty separately from control challenges during training. Depth images are usually costly to render in the simulation and to process on board. Inspired by the training idea of using waypoints as the locomotion command [52], we choose the relative position in the robot frame as the most compact representation of the target object, which provides adequate information to track the goal. We provide the desired velocity, as determined by the operator. The hope is, that the network learns to control the quadrupedal robot to track toward the object, following the given velocity value.

A. Goal Oriented Reward

During the RL training process, to motivate the robot to be agile and precise, we switch between two reward functions. We set the first tracking reward term r_{vel} similarly to [6], based on the velocity command, encouraging the robot to follow the velocity command running toward the goal in an agile fashion. Additionally, we use the tracking yaw $r_{\text{tracking yaw}}$ reward term to help the robot face towards the goal.

In order for the robot to learn to also be precise, i.e. to accurately catch the target object, we set the r_{pos} tracking reward based on the relative goal position in the robot end-effector frame, which exponentially increases with decreasing target-end-effector distance. We add a constant 1 to make $r_{\text{pos}} \geq r_{\text{vel}}$, such that the robot wouldn't fall into some local optima while switching between the two reward functions. Finally, a bit-wise reward indicates whether we successfully caught the object.

$$r_{\text{tracking goal}} = \begin{cases} r_{\text{vel}} = \min\left(\left\langle \mathbf{v}, \hat{\mathbf{d}}_w \right\rangle, v_{cmd}\right), & d_{xy} > D \\ r_{\text{pos}} = \exp\left(-\left\|\mathbf{p}-\mathbf{x}\right\|/\alpha\right)+1, & d_{xy} \leq D \end{cases} \tag{1}$$

$$r_{\text{tracking yaw}} = \exp\left(-|y_{goal}-y|/\sigma\right) \tag{2}$$

$\hat{\mathbf{d}}_w = \frac{\mathbf{p_{xy}}-\mathbf{x_{xy}}}{\|\mathbf{p_{xy}}-\mathbf{x_{xy}}\|}$ denotes the 2D goal direction vector in the xy-plane, x_{xy} is the robot base position and p_{xy} is the goal position in the xy-plane in the world frame. $\mathbf{v}$ is the 2D base linear velocity in the horizontal plane and v_{cmd} is the scalar velocity value we command. Regarding r_{pos}, x is the 3D robot base position and p is the goal position in the world frame. d_{xy} denotes the distance between the robot end effector and the target ball in the xy-plane, and D is the threshold set to switch the tracking reward. y_{goal} is the target global yaw angle and y is the current global yaw angle of the robot base. α and σ are scaling factors.

So r_{vel} is rewarding the training to quickly move the robot towards the general goal direction, while r_{pos} takes over when the robot is close to the goal object and strongly rewards high accuracy w.r.t. gripper-target distance by scaling expo-

nentially, thus together training the network to be agile **and** precise.

We also use the regularization terms, including conserving mechanical energy as in [5] to encourage reasonable biomechanically optimal gait. To overcome the gap of exploration, we set an easily-deployed curriculum for goal height. At the very beginning, the robot can easily capture the target balls almost on the same level as the base. As the robot get more successful (catching the ball and not falling over), our code will gradually choose to spawn more episodes with more difficult/ higher target positions, requiring the robot to learn to jump. However, it still takes some time for the robot to learn how to jump up and grab objects. This is because, at the beginning, the rewards obtained by the robot trying to jump up and grab the object were relatively sparse, so sometimes, the rewards obtained by stopping directly below the object reach a local optima. This can be solved by balancing the rewards for grabbing objects and distance rewards.

B. Collision Shape and Grasping System in Simulation

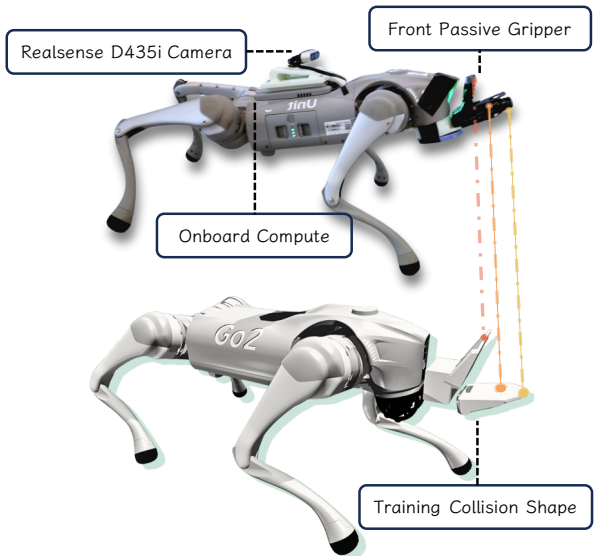

Fig. 3: **Hardware Setup.** Maintaining consistency in the collision shape of the front gripper between simulation and reality helps the robot learn to correctly trigger contact with the ball using the gripper at the appropriate position.

a) Collision Shape Setup: The gripper on our actual robot is triggered via a passive mechanism to close when encountering a force induced by the ball hitting the gripper. For this reason, and to simplify the simulation, we do not simulate a gripper with an active joint that is cathing the ball by physically restraining it between the jaws. Instead, we use a simple catch PD controller to "catch" the ball in the mouth once it is very close to the center mouth point.

We apply mutual external forces between the ball and the gripper with a strong PD controller so that the gripper can hold the ball firmly once it reaches the correct grip, i.e. $\|p_{\text{ball}} - p_{\text{gripper}}\| < D_{\text{grasp}}$.

$$F_{\text{grasp}} = (K_{\text{p, grasp}}(p_{\text{ball}} - p_{\text{gripper}}) - K_{\text{d, grasp}}(v_{\text{ball}} - v_{\text{gripper}})) * \mathbb{1}[\|p_{\text{ball}} - p_{\text{gripper}}\| < D_{\text{grasp}}] \tag{3}$$

The mutual external force to the gripper is described as Equation 3, where p_{ball} is the ball position in the world frame and p_{gripper} is the middle point of the gripper in the world frame. The external force applied to the ball is the negative of F_{grasp}. To enforce a firm grasp, we set $K_{\text{p, grasp}} = 150.0$, $K_{\text{d, grasp}} = 2.0$.

b) Ball Flying Away vs. Not Flying Away: Due to the high-speed dynamics of our task, in the simulation the ball tends to fly away and will never be caught once it collides with the gripper on the robot - even if it is colliding on the inside faces of the mouth. But we know from our real robot experiment, that the gripper will catch the ball even if it comes into the mouth with great velocity. Thus, we apply another correctional force to the ball with the catch PD controller to prevent the ball from flying away after being hit.

Another problem is, that a collision of the ball with other parts of the robot model will caues it to fly far away. In order to keep the ball close, as if it would be hanging from a rope, we slowly move the ball back to its original position using a different PD controller. By tuning the stiffness and damping parameters of this controller, we can adjust the difficulty for the robot as well: In the beginning the ball comes back quickly, and later very slowly. In our experiments we have seen that this way the robot can learn its own collision information and better avoid collisions.

C. System Pipeline

After Reinforcement Learning using the rewards and methods outlined above, our system uses the trained policy network to convert observation inputs into goal joint angles, which the robot's PD motor controller then uses to calculate and apply motor torques to the real robot.

a) Target localization: To detect the target, we utilize the RGB-D camera mounted on the back of the quadrupedal robot for exteroception, which captures images at 30 Hz. To simplify the training of the policy network and enhance its generalization, we represent the target information in the observation as the coordinate position of the object in the end-effector's frame. We first obtain the pixel coordinates of the target from the image, and then combine this with depth information and the camera's intrinsic parameters to derive the object's coordinates in the camera frame. Finally, using the camera's extrinsic parameters and its pose relative to the end-effector, we can calculate the target object's coordinate position in the end-effector frame. Additionally, there is a one-bit parameter that indicates whether the object has been detected. For training the simulator checks if the object is in the field of view of an assumed virtual camera to decide if the object pose is provided or not.

Many off-the-shelf object detection methods could be used. However, we use the simplest HSV (Hue, Saturation, Value) detector in our real-world experiments, due to the

real-time requirement and it working sufficiently well for our simple use-case. We extract the object's position within the image frame by utilizing openCV's "findContours" function [53]. We then select the median depth value of all ranges in the biggest contour. Due to the high-speed motion of the robot, the collected images are prone to motion blur, resulting in the estimated contour not well overlapping with the pixels of the ball in the depth image, such that sometimes the calculated median depth is not on the ball but on the background. Therefore, we filter the estimated target distances of consecutive frames for big jumps in the distance.

b) Inference Network: Considering that the dog may lose tracking with the ball while jumping and the latency in obtaining the target position, we employ a memory-equipped network, specifically a GRU (Gated Recurrent Unit), which processes the hidden state followed by a simple three-layer MLP (Multi-Layer Perceptron).

c) Deployment: Our deployment on the real robot is based on ROS2 (Robot Operating System 2), which utilizes DDS (Data Distribution Service) for communication. ROS2 serves as a bridge connecting the policy network, robot node, and external sensors, enabling seamless integration of various modules and simplifying the collection of information from multiple sensors. The policy network converts the observation input, which includes proprioception and goal position coordinates obtained from the object detector, into goal joint angles at a frequency of 50 Hz. Subsequently, the PD motor controller calculates the motor torques based on the goal joint angles, current joint angles, and joint velocities, applying them to the real robot.

$$\tau = K_p * (q_{target} - q) - K_d * v \tag{4}$$

where q_{target} and q are the target joint angles and current joint angles, v denotes current joint velocities, and we tuned the parameters to $K_p = 35$ and $K_d = 0.6$. To ensure safe deployment, we implement a safety mechanism that terminates the program when the joint angles exceed the safe range or the angular velocity exceeds 55 degrees per second. The HSV-based object detector runs at the camera frame rate of $30Hz$.

IV. EXPERIMENTS

In order to validate our method and test the limits of our framework, we propose several experiments. We first investigate the necessity of the memory mechanism in our task using a trivial MLP, a GRU based network (our method), and a Transformer-based network. Then, we test how well our system estimates the target position by providing the truth value of the target height. By comparing these two setups, we evaluate the possible improvement of estimating the target position, as this is difficult while the robot is in motion (see below).

A. Experiment Setup

We deploy our policy on a Unitree Go2 robot with 12 joints and one extra mouth-like gripper with a total weight of 15kg. The height of the standing base is about 30cm and its body length is about 70cm. We run the policy on a Jetson Orin NX mounted on the robot. For detecting the target, we use an Intel RealSense D435i mounted on the back of the quadrupedal robot for exteroception with a frame rate of 30Hz. Since our focus is not on hardware and our task requires the gripper to be able to instantly close and grasp objects, we design a passive gripper with strong magnets that can be swiftly triggered.

B. Simulation Experiments

Policy Backbone	Success Rate					
	h (Simulation)			h (Real-World)		
	0.5m	**0.7m**	**0.9m**	**0.5m**	**0.7m**	**0.8m**
MLP	0.819	0.354	0.000	-	-	-
GRU	0.922	0.708	0.433	9/10	3/10	1/10
Transformer	0.870	0.711	0.437	8/10	3/10	2/10

TABLE I: **Success Rate Experiments.** We quantitatively test the success rates of three policy networks with MLP, GRU, and Transformer as backbones for target heights of 0.5m, 0.7m, and 0.9m/ 0.8m in simulation/ real-world environments.

1) Success rate: A robots's attempt is classified as success if it jumps, catches the target ball, and then lands without falling over. We consider it successful if the robot catches the ball within three attempts of its front legs leaving the ground, as the robot sometimes requires real-time adjustment information to better know the desired jumping height. We test the success rate of the robot at different target heights from 0.5m to 0.9m in the simulation environment, as shown in Table I. We also present keyframe animations of the robot grasping different target objects in Figure 1. Due to the significant impact of the allowable torque on the robot's jumping ability in the simulation, failing to impose a torque limit could result in a large sim-to-real gap, making the success rates in the simulation less meaningful. Thus, we set a maximum torque limit of 30 Nm for the robot in the simulation. We selected the checkpoints of the three best-performing corresponding policy networks for testing. The policy networks based on MLP, GRU, and Transformer achieve success rates of over 80% at a target height of 0.5m, with MLP performing the worst. However, when the target height increases to 0.7m, the success rate decreases as the task difficulty rises, with MLP dropping to around 35%, while GRU and Transformer maintained success rates above 70%. As the target height further increases, the MLP struggles due to its lack of memory, resulting in failure at 0.9m, while both the GRU and Transformer achieve success rates exceeding 40%. The overall performance of the GRU and Transformer is quite similar. During the training process, we found that as the required grabbing height increases from a level that did not require jumping to a height that did, a lack of careful adjustment of the reward could result in the model failing to learn that it could jump, causing it to get stuck in a local optimum. The policy based on the Transformer architecture demonstrates some advantages in escaping from this local optimum.

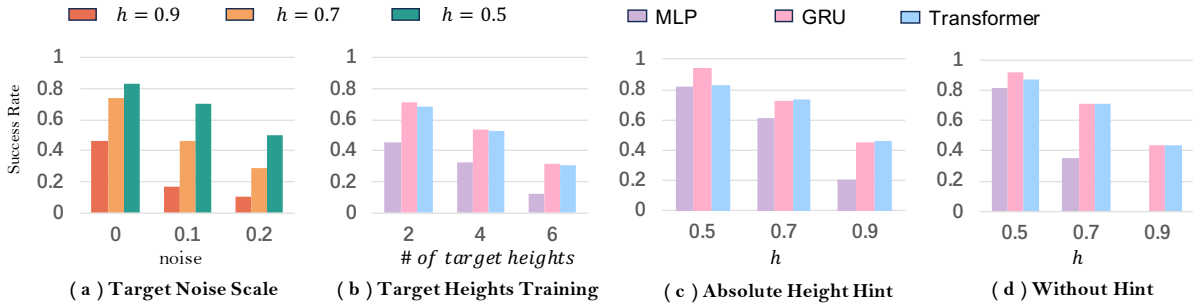

Fig. 4: **Comparisons.** (a) We examine the success rate of GRU for catching the target object while introducing uniform noise to the goal position at various heights. (b) We analyze the success rate of the training range for the number of different target heights during training across three different backbone architectures. (c) and (d) We evaluate the success rate based on whether the input includes the absolute target height, again comparing the performance of the three backbone models.

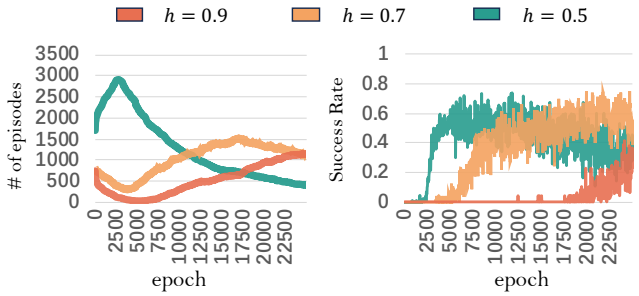

Fig. 5: **Curriculum Learning.** When broadening the height range of objects that robots can grasp through curriculum settings, we observe that the success rate for a specific height may decline due to a reduced number of episodes associated with that height.

2) Curriculum Learning: During the training process, we use a curriculum to adjust the height of the target objects, preventing the agent from giving up on catching the ball when faced with difficult targets. When expanding the height range of objects grasped by robots by curriculum setting, we noticed that the success rate of a specific height may decrease due to the decrease in the number of environments corresponding to this height. In one training session shown in Figure 5, we set up 4096 parallel environments and applied height curriculum to dynamically adjust the task difficulty. During the training process, while the number of environments set to a height of 0.5m decreases to 25% of the total number of environments, the success rate at that height exhibits a downward trend.

As shown in Figure 4 (b), as the number of different heights that the network is trained to master increases, the average success rate goes down. We then hypothesize that this may result from the robot's potential pitch during movement, which makes the representation of the target's height using relative position less clear. Therefore, we introduce a hint of the target object's absolute height into the robot's observation input to compare the impact of this hint on the success rate. Figures 4 (c) and (d) illustrate the two scenarios with and without the absolute height hint. However, the inclusion of the hint significantly impacts the success rate only for the MLP backbone at heights above 0.7m, which may be attributed to MLP's lack of memory capability. We speculate that this may also be related to the different gaits the robot exhibits when grasping objects at varying heights, as shown in Figure 1. After the curriculum updates, training on a new target may have affected the model's performance on the original target. This is also a point we can explore further in our future work.

C. Real-World Experiments

We also quantitatively test our policy in the real world. We select the best-performing checkpoints based on GRU and Transformer backbones from the simulator for testing on the Unitree Go2 real robot. The success rates of different heights of the target ball over 10 trials each are shown in Table I. When the target object's height is 0.5m - just slightly above the robot's standing height - the success rate of grasping the object is very high, more than 80%. However, as the height of the object increases, the difficulty of the task significantly rises, leading to a corresponding decrease in the success rate.

Due to the significant noise and latency in the relative position information of the target object captured by the camera while the robot is in high-speed motion, the success rate in the real world is lower compared to experiments at the same height conducted in the simulation. When the target height is 0.7m, the robot's success rate is approximately 30%; however, when the target height increases to 0.8m, the success rate drops to 10%.

There exists a sim-to-real gap, as the real robot cannot achieve the highest height successfully reached in the simulation. The real robot can catch the highest ball at 0.8m, roughly twice the height at which the robot is standing. During the experiments, we found that minimizing the difference in camera latency between the real world and training, by adding a delay to providing the simulated object pose, does improve the success rate of target grabbing. The model that performs best on the real robot has a latency sample range of [0.03, 0.08] seconds during the training process.

We test our policy in several different indoor and outdoor environments, including relatively smooth surfaces, grassy areas, and uneven outdoor stone pathways. The robot can catch a fixed ball or play with a moving ball held by a human, demonstrating the ability to continuously follow a mid-air ball moving at speeds of up to 3m/s with direction changes, maintaining a close pursuit without losing track of the target, as shown in Figure 1 and the video.

V. CONCLUSION, LIMITATION AND FUTURE WORK

In this work, we demonstrate the capability of a quadruped robot equipped with an end-effector designed to mimic a dog's mouth to jump and catch small objects. By training the robot in simulation, we achieve notable performance, allowing it to capture targets at various heights in the simulation, and with the highest height of 0.8m real-world deployment. Based on the experiments, we show that the robot exhibits different gaits when grasping objects at different height levels. Additionally, there is a noticeable gap between the results in real-world scenarios and those in simulation. This discrepancy may stem from hardware limitations, the sim-to-real gap, and noise from the existing detector during high-speed motion. These factors represent key areas for future research aimed at enhancing the robot's performance in high-dynamic movements.

We also did some preliminary work to enable the robot pursue and catch arbitrary objects using the Grounding DINO [54] open-vocabulary detector and Mobile SAM [55] tracker. This pipeline works on the Nvidia Jetson NX at 8 ± 2 Hz. Although this frequency is not very sufficient for the robot to accurately catch when needing to jump up, it shows our ability to track different objects and play with humans. This indicates that our agile locomotion skills can be integrated with the system based on vision-language models (VLMs), such as in a helping task [4], allowing us to perform a broader range of tasks, which we will pursue in the future.

ACKNOWLEDGMENT

This project is supported by Shanghai Qi Zhi Institute and ONR grant N00014-20-1-2675 and has been partially funded by the Shanghai Frontiers Science Center of Human-centered Artificial Intelligence. The experiments of this work were supported by the core facility Platform of Computer Science and Communication, SIST, ShanghaiTech University. Thanks to Qi Wu for the detailed technical support. We thank Zipeng Fu, Xuxin Cheng, Wenhao Yu and Erwin Coumans for the feedback and discussion.

REFERENCES

[1] Aftab E Patla. Understanding the roles of vision in the control of human locomotion. *Gait & posture*, 1997.

[2] Xiaoyu Huang, Zhongyu Li, Yan-Ling Xiang, Yiming Ni, Yufeng Chi, Yunhao Li, Lizhi Yang, Xue Bin Peng, and Koushil Sreenath. Creating a dynamic quadrupedal robotic goalkeeper with reinforcement learning. *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2715–2722, 2022.

[3] Benedek Forrai, Takahiro Miki, Daniel Gehrig, Marco Hutter, and Davide Scaramuzza. Event-based agile object catching with a quadrupedal robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12177–12183, 2023.

[4] Qi Wu, Zipeng Fu, Xuxin Cheng, Xiaolong Wang, and Chelsea Finn. Helpful doggybot: Open-world object fetching using legged robots and vision-language models. In *arXiv*, 2024.

[5] Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher G Atkeson, Sören Schwertfeger, Chelsea Finn, and Hang Zhao. Robot parkour learning. In *Conference on Robot Learning CoRL*, 2023.

[6] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450, 2024.

[7] Christian Gehring, Stelian Coros, Marco Hutter, Carmine Dario Bellicoso, Huub Heijnen, Remo Diethelm, Michael Bloesch, Peter Fankhauser, Jemin Hwangbo, Mark Hoepflinger, and Roland Siegwart. Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot. *IEEE Robotics & Automation Magazine*, 23(1):34–43, 2016.

[8] Boston dynamics: Atlas.

[9] Quan Nguyen, Matthew J Powell, Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Optimized jumping on the mit cheetah 3 robot. In *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[10] Matt Zucker, J. Andrew Bagnell, Christopher G. Atkeson, and James Kuffner. An optimization approach to rough terrain locomotion. In *2010 IEEE International Conference on Robotics and Automation*, pages 3589–3595, 2010.

[11] Rika Antonova, Akshara Rai, and Christopher G. Atkeson. Deep kernels for optimizing locomotion controllers. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 47–56. PMLR, 13–15 Nov 2017.

[12] Matthew Zucker, Nathan D. Ratliff, Martin Stolle, Joel E. Chestnutt, J. Andrew Bagnell, Christopher G. Atkeson, and James Kuffner. Optimization and learning for rough terrain legged locomotion. *Int. J. Robotics Res.*, 30(2):175–191, 2011.

[13] J. Zico Kolter and A. Ng. The stanford littledog: A learning and rapid replanning approach to quadruped locomotion. *The International Journal of Robotics Research*, 30:150 – 174, 2011.

[14] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry, and Stefan Schaal. Fast, robust quadruped locomotion over challenging terrain. In *2010 IEEE International Conference on Robotics and Automation*, pages 2665–2670, 2010.

[15] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

[16] Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, Maurice Fallon, and Ioannis Havoutis. Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control. *IEEE Transactions on Robotics*, 38(5):2908–2927, 2022.

[17] Young-Ha Shin, Seungwoo Hong, Sangyoung Woo, JongHun Choe, Harim Son, Gijeong Kim, Joon-Ha Kim, KangKyu Lee, Jemin Hwangbo, and Hae-Won Park. Design of kaist hound, a quadruped robot platform for fast and efficient locomotion with mixed-integer nonlinear optimization of a gear train. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6614–6620, 2022.

[18] Hae-Won Park, Patrick M Wensing, Sangbae Kim, et al. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. *RSS*, 2015.

[19] Chuong Nguyen, Lingfan Bao, and Quan Nguyen. Continuous jumping for legged robots on stepping stones via trajectory optimization and model predictive control. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 93–99. IEEE, 2022.

[20] Gabriel Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. In *Robotics: Science and Systems*, 2022.

[21] Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 2022.

[22] Zipeng Fu, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Minimizing energy consumption leads to the emergence of gaits in legged robots. In *CoRL*, 2021.

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

[24] Gabriel B Margolis, Tao Chen, Kartik Paigwar, Xiang Fu, Donghyun Kim, Sangbae Kim, and Pulkit Agrawal. Learning to jump from pixels. *CoRL*, 2021.

[25] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, October 2020.

[26] I Made Aswin Nahrendra, Byeongho Yu, and Hyun Myung. Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5078–5084, 2023.

[27] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In *6th Annual Conference on Robot Learning*, 2022.

[28] Ruihan Yang, Ge Yang, and Xiaolong Wang. Neural volumetric memory for visual locomotion control. *CVPR*, 2023.

[29] Antonio Loquercio, Ashish Kumar, and Jitendra Malik. Learning visual locomotion with cross-modal supervision. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7295–7302, 2023.

[30] David Hoeller, Nikita Rudin, Christopher Choy, Animashree Anandkumar, and Marco Hutter. Neural scene representation for locomotion on structured terrain. *IEEE Robotics and Automation Letters*, 2022.

[31] Nikita Rudin, David Hoeller, Marko Bjelonic, and Marco Hutter. Advanced skills by learning locomotion and local navigation end-to-end. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2497–2503, 2022.

[32] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, 2022.

[33] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 2019.

[34] Chenhao Li, Marin Vlastelica, Sebastian Blaes, Jonas Frey, Felix Grimminger, and Georg Martius. Learning agile skills via adversarial imitation of rough partial demonstrations. In *Conference on Robot Learning*, 2022.

[35] Laura M. Smith, J. Chase Kew, Tianyu Li, Linda Luu, Xue Bin Peng, Sehoon Ha, Jie Tan, and Sergey Levine. Learning and adapting agile locomotion skills by transferring experience. In *Robotics: Science and Systems*, 2023.

[36] Junfeng Long, Wenye Yu, Quanyi Li, Zirui Wang, Dahua Lin, and Jiangmiao Pang. Learning h-infinity locomotion control, 2024.

[37] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, P. Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv: Learning*, 2018.

[38] Ken Caluwaerts, Atil Iscen, J. Chase Kew, Wenhao Yu, Tingnan Zhang, Daniel Freeman, Kuang-Huei Lee, Lisa Lee, Stefano Saliceti, Vincent Zhuang, Nathan Batchelor, Steven Bohez, Federico Casarini, José Enrique Chen, Omar Andrés Carmona Cortes, Erwin Coumans, Adil Dostmohamed, Gabriel Dulac-Arnold, Alejandro Escontrela, Erik Frey, Roland Hafner, Deepali Jain, Bauyrjan Jyenis, Yuheng Kuang, Edward Lee, Linda Luu, Ofir Nachum, Kenneth Oslund, Jason Powell, Diego M Reyes, Francesco Romano, Feresteh Sadeghi, R. J. Sloat, Baruch Tabanpour, Daniel Zheng, Michael Neunert, Raia Hadsell, Nicolas Manfred Otto Heess, Francesco Nori, Jeffrey A. Seto, Carolina Parada, Vikas Sindhwani, Vincent Vanhoucke, and Jie Tan. Barkour: Benchmarking animal-level agility with quadruped robots. *ArXiv*, abs/2305.14654, 2023.

[39] Fabian Jenelten, Junzhe He, Farbod Farshidian, and Marco Hutter. Dtc: Deep tracking control. *Science Robotics*, 9(86):eadh5401, 2024.

[40] Elena Arcari, Maria Vittoria Minniti, Anna Scampicchio, Andrea Carron, Farbod Farshidian, Marco Hutter, and Melanie N Zeilinger. Bayesian multi-task learning mpc for robotic mobile manipulation. *IEEE Robotics and Automation Letters*, 8(6):3222–3229, 2023.

[41] Huy Ha, Yihuai Gao, Zipeng Fu, Jie Tan, and Shuran Song. Umi on legs: Making manipulation policies mobile with manipulation-centric whole-body controllers, 2024.

[42] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, 2022.

[43] Tifanny Portela, Gabriel B Margolis, Yandong Ji, and Pulkit Agrawal. Learning force control for legged manipulation. *arXiv preprint arXiv:2405.01402*, 2024.

[44] Guoping Pan, Qingwei Ben, Zhecheng Yuan, Guangqi Jiang, Yandong Ji, Jiangmiao Pang, Houde Liu, and Huazhe Xu. Roboduet: A framework affording mobile-manipulation and cross-embodiment. *arXiv preprint arXiv:2403.17367*, 2024.

[45] Hiroshi Ito, Kenjiro Yamamoto, Hiroki Mori, and Tetsuya Ogata. Efficient multitask learning with an embodied predictive model for door opening and entry with whole-body control. *Science Robotics*, 2022.

[46] Xuxin Cheng, Ashish Kumar, and Deepak Pathak. Legs as manipulator: Pushing quadrupedal agility beyond locomotion. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[47] Zhengmao He, Kun Lei, Yanjie Ze, Koushil Sreenath, Zhongyu Li, and Huazhe Xu. Learning visual quadrupedal loco-manipulation from demonstrations. *arXiv preprint arXiv:2403.20328*, 2024.

[48] Philip Arm, Mayank Mittal, Hendrik Kolvenbach, and Marco Hutter. Pedipulate: Enabling manipulation skills using a quadruped robot's leg. *ArXiv*, abs/2402.10837, 2024.

[49] Yandong Ji, Gabriel B. Margolis, and Pulkit Agrawal. Dribblebot: Dynamic legged manipulation in the wild. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5155–5162, 2023.

[50] 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. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1479–1486, 2022.

[51] Changyi Lin, Xingyu Liu, Yuxiang Yang, Yaru Niu, Wenhao Yu, Tingnan Zhang, Jie Tan, Byron Boots, and Ding Zhao. Locoman: Advancing versatile quadrupedal dexterity with lightweight loco-manipulators, 2024.

[52] Chong Zhang, Nikita Rudin, David Hoeller, and Marco Hutter. Learning agile locomotion on risky terrains. *arXiv preprint arXiv:2311.10484*, 2023. Using goal point as command interface.

[53] Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2015.

[54] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision – ECCV 2024*, pages 38–55, Cham, 2025. Springer Nature Switzerland.

[55] Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023.