Robust Robot Walker: Learning Agile Locomotion over Tiny Traps

Shaoting Zhu[1,2], Runhan Huang[1,2], Linzhan Mou[2,3], Hang Zhao†[1,2]

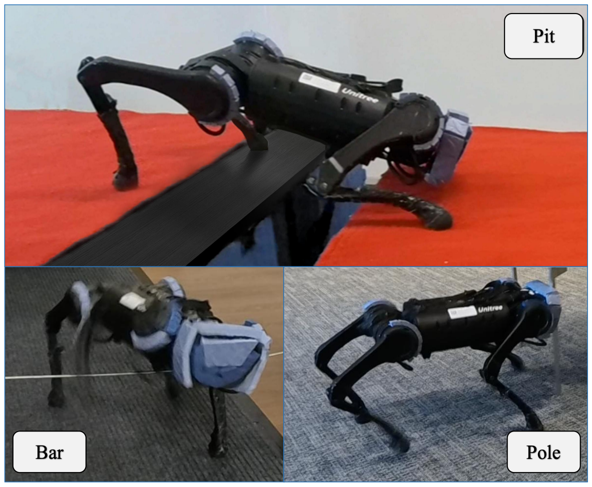

Fig. 1: Our `Robust Robot Walker` is passing through various challenging "tiny traps" including Pit, Bar, and Pole solely relying on its proprioception by single policy.

***Abstract*— Quadruped robots must exhibit robust walking capabilities in practical applications. In this work, we propose a novel approach that enables quadruped robots to pass various small obstacles, or "tiny traps". Existing methods often rely on exteroceptive sensors, which can be unreliable for detecting such tiny traps. To overcome this limitation, our approach focuses solely on proprioceptive inputs. We introduce a two-stage training framework incorporating a contact encoder and a classification head to learn implicit representations of different traps. Additionally, we design a set of tailored reward functions to improve both the stability of training and the ease of deployment for goal-tracking tasks. To benefit further research, we design a new benchmark for tiny trap task. Extensive experiments in both simulation and real-world settings demonstrate the effectiveness and robustness of our method. Project Page: https://robust-robot-walker.github.io/.**

I. INTRODUCTION

Humans and animals have the ability to walk robustly in complex environments, relying on proprioception to avoid various obstacles such as strings, poles, and ground pits. However, these same challenges present significant difficulties for robots. In real-world scenarios, seemingly minor traps can severely impact a robot's mobility. Many obstacles are tiny or positioned below or behind the robot, making them difficult to detect with external sensory devices like depth cameras, as shown in Fig. 2. Small objects like narrow bars or poles often produce unreliable data in the depth images, appearing intermittently noisy or as a zero missing

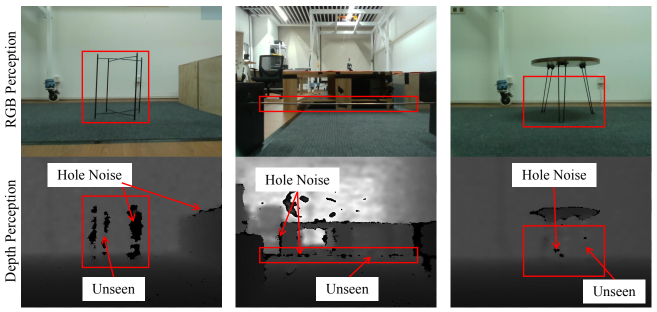

Fig. 2: Perception failure in the tiny trap scenarios.

value, making them indistinguishable from the hole noise of other obstacle's edges. Additionally, since realistic RGB images cannot be accurately rendered in simulations, using them in real-world is limited due to a significant sim-to-real gap. This perception failure raises the need for developing control policies that enable robots to overcome such trap-type obstacles without relying on additional sensory equipment.

Learning agile locomotion over tiny traps presents several challenges. First, frameworks [1], [2] that rely on exteroceptive inputs are ineffective for tasks involving tiny traps, as both RGB and depth images are difficult to leverage in these scenarios. Second, with incomplete perceptual information, it is difficult to learn a blind walking policy directly from scratch. Some privileged information is required to guide the training. Lastly, while some goal-tracking frameworks [3], [4] have been proposed, they often lack omnidirectional movement capabilities or rely heavily on external localization techniques. Moreover, these frameworks frequently employ sparse rewards, which cause the instability of training and complicate real-world deployment.

To address the challenges of learning agile locomotion over tiny traps using proprioception, we propose a novel solution with several key contributions.

- First, we introduce a **two-stage training framework** that **relies solely on proprioception**, enabling a robust policy that successfully passes tiny traps in both simulation and real-world environments.
- Second, we develop an **explicit-implicit dual-state estimation paradigm** to better learn the latent space, and reduce the sim-to-real gap.
- Third, we **redefine the task as goal tracking**, and incorporate carefully designed **dense reward functions and fake goal commands**. This approach achieves approximate omnidirectional movement in real-world.
- Finally, we introduce **a new benchmark for tiny trap tasks** and conduct extensive experiments in both simulation and real-world scenarios, demonstrating the robustness and effectiveness of our method.

[1]IIIS, Tsinghua University, Beijing, China
[2]Shanghai Qi Zhi Institute, Shanghai, China
[3]GRASP Lab, University of Pennsylvania, Philadelphia, PA, USA
† Corresponding author. E-mail: `hangzhao@mail.tsinghua.edu.cn`

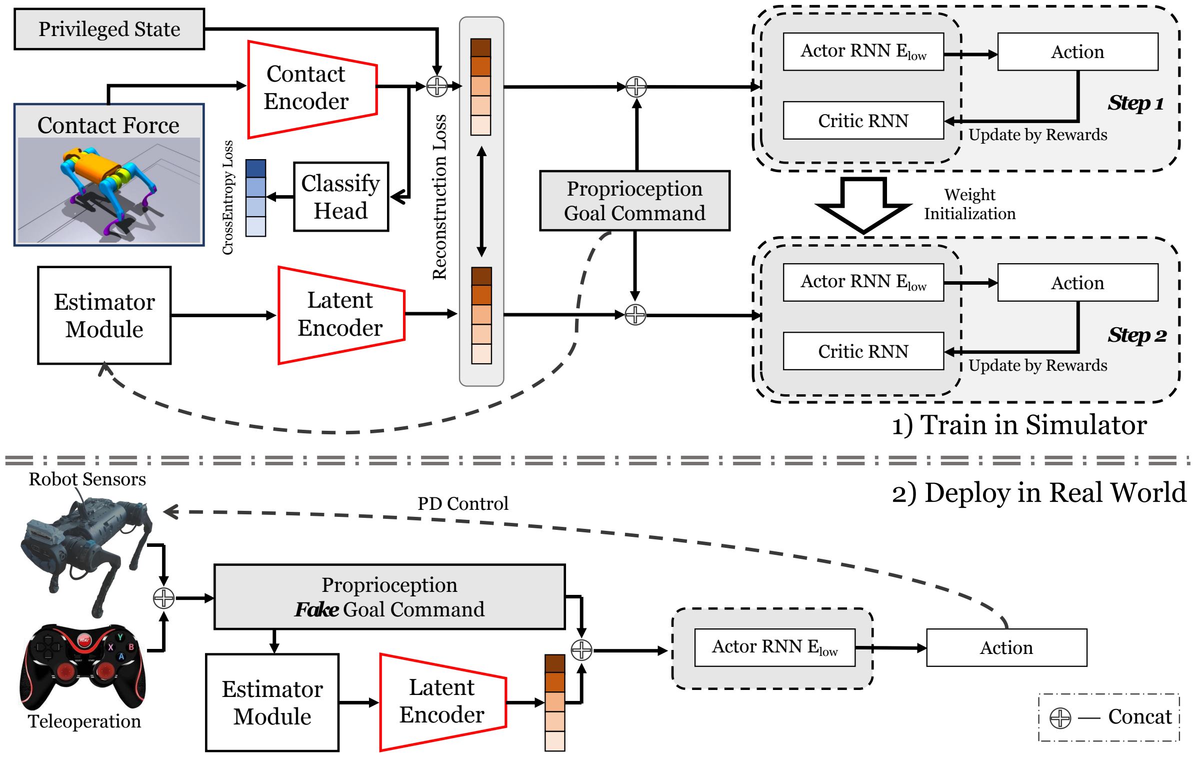

Fig. 3: The training and deployment overview of `Robust Robot Walker`. Our method achieves tiny trap passing and approximate omnidirectional movement. Each color on the quadruped robot corresponds to a type of joint link: Orange-Base, Yellow-Hip, Blue-Thigh, Purple-Calf, and Foot (Unseen).

II. RELATED WORKS

A. Learning-Based Locomotion

The deep reinforcement learning (DRL) paradigm has demonstrated its ability to learn legged locomotion behaviors in simulators [5]. Unlike model-based control methods [6], DRL enables robots to handle corner cases in simulation through an end-to-end learning process, resulting in robust, transferable policies. These simulation-trained policies allow robots not only to walk on flat terrain [7], but also to achieve high-speed running [8], [9], traverse muddy or uneven terrains [10], stand on rear legs [11], [12], open doors [11], climb stairs [13], [14], scale rocky terrain [4], [15], and even perform high-speed parkour [2], [3], [16]–[18].

B. Collision Detection

Collisions in robots can be classified into several stages: the pre-collision phase [19]–[21], collision detection phase [19], [22], [23], collision isolation phase [19], [24], collision identification phase [19], [24], [25], collision classification phase [26], collision reaction phase [27]–[29], and post-collision phase [26], [29]. Collision detection methodologies for quadruped robots are generally divided into model-based and model-free approaches. Model-based methods typically employ state estimation techniques [30]–[32]. Some approaches leverage exteroceptive sensors [33], while others rely purely on proprioception for estimation [34]. Model-free methods, on the other hand, involve training neural network-based contact estimators through deep reinforcement learning, which can be either implicit [35] or explicit [15].

III. METHOD

A. Task Definition

The task is defined as passing through a set of tiny traps, denoted as $\mathcal{T}$. An onboard camera typically fails to detect such traps. These traps are categorized into three main types: *Bar*, *Pit*, and *Pole*, as illustrated in Fig. 4. A *Bar* refers to a horizontal thin bar positioned over a plane at a height below the quadruped robot's head. A *Pit* is a small depression between two planes that can cause the robot's legs to slip, and it is not visible from the normal front view. A *Pole* is a thin, upright pole standing on a plane. When encountering these traps, the robot is provided with `constant` control commands and must adjust its speed autonomously to pass them. Note that in training and demonstrations, the traps include both thin and thick *Bars*/*Poles*. While the onboard camera can detect thicker ones, they are included only to showcase the generalization capabilities of our control policy.

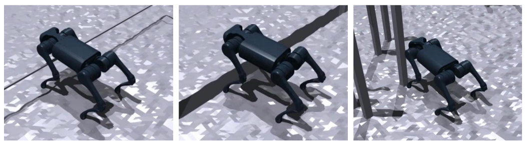

Fig. 4: Three categories of tiny traps: Bar, Pit, and Pole.

B. Reinforcement Learning Setting

We decompose the locomotion control problem into discrete locomotion dynamics, with a discrete time step $d_t = 0.02s$. We use the Proximal Policy Optimization algorithm (PPO) [36] to optimize our policy. Inspired by [3], [4], we formulate the problem as goal tracking instead of velocity tracking.

State Space: The entire process includes the following four types of observation: proprioception $\boldsymbol{p}_t$, privileged state $\hat{\boldsymbol{s}}_t$, contact force $\hat{\boldsymbol{c}}_t$, and goal command $\boldsymbol{g}_t$. **1) Proprioception $\boldsymbol{p}_t$** contains gravity vector and base angular velocity from IMU, joint positions, joint velocities, and last action. **2) Privileged state $\hat{\boldsymbol{s}}_t$** contains base linear velocity (unreliable from IMU) and the ground friction. **3) Contact force $\hat{\boldsymbol{c}}_t$** includes contact force with environment meshes of each joint link, refer to the robot link in different colors in Fig. 3. Each contact force is clipped, and normalized to $[-1,1]$. **4) Goal command $\boldsymbol{g}_t$** includes goal position $\Delta G = (\Delta x, \Delta y, \Delta z)$ relative to the current robot frame, and the time remaining to complete Δt. At the beginning of one episode, we randomly sample a goal position fixed in the world frame and set Δt to the episode length. In every time step, we calculate ΔG based on the robot pose, and update Δt to the remaining time of the current episode. Since our task does not involve height change, Δz is always set to zero. To facilitate the training of a standstill state, we set $\Delta G = (0,0,0)$ when $\|\Delta G\|_2 < 0.2$.

Action Space: The action space $\boldsymbol{a}_t \in \mathbb{R}^{12}$ is the desired joint positions of 12 joints.

C. Reward Function

The reward function has three components: task reward r_t^T, regularization reward r_t^R, and style reward r_t^S. The total reward is the sum of three items: $r_t = r_t^T + r_t^R + r_t^S$.

1) Task reward. r_{goal}, $r_{heading}$, and r_{finish} are the main components. Unlike previous works, we define the r_{goal} as a dense reward throughout the entire episode. In our setup, the robot can only perceive tiny traps upon contact, meaning it cannot predict and avoid them from a distance against the reward function. This eliminates the need for free movement exploration as seen in [4], and significantly enhances the stability of the training process.

$$r_{goal} = \frac{1}{0.4 + \|\Delta G\|_2}, \tag{1}$$

We aim for the robot to always move toward the goal. Without the heading reward, it may encounter traps horizontally or backward, which is equivalent to multi-task learning, and greatly increases the difficulty of training. Although heading reward allows the robot to only pass through the trap in a straight or with a small angle, it greatly speeds up convergence and improves the passing success rate. $\epsilon = 10^{-6}$.

$$r_{heading} = \begin{cases} \dfrac{\Delta x}{\|\Delta G\|_2 + \epsilon}, & \|\Delta G\|_2 \neq 0 \\ 1, & \|\Delta G\|_2 = 0 \end{cases}, \tag{2}$$

Besides, the policy should be able to operate stably for a long time on the real robot. Therefore, we introduce a finish reward to encourage the robot to stand still when near the goal. When $\Delta G = (0,0,0)$:

$$r_{finish\_pos} = \sum_{i=1}^{12} |q - q_{default}|, \tag{3}$$

$$r_{finish\_vel} = \|v\| + \|\omega\|, \tag{4}$$

$$r_{finish} = \lambda_1 \cdot r_{finish\_pos} + \lambda_2 \cdot r_{finish\_vel}. \tag{5}$$

2) Regularization reward. Regularization reward is designed to make the robot move smoothly, safely, and naturally. A key component of this is the velocity limit reward. Unlike previous work, our goal is not for the robot to reach the speed limit, which is unsafe in a real deployment.

$$r_{vel\_limit} = (\omega_z < \omega_{limit}) \cdot (\|v_{x,y}\|_2 < v_{limit}) \tag{6}$$

Other regularization rewards include stall, leg energy, dof vel, etc. The details can be found in *Appendix A*.

3) Style reward. We use the Adversarial Motion Priors (AMP) style reward to gain a natural gait and speed up convergence following [13], [37].

D. Training and Deployment

1) Training. To effectively learn the privileged state and improve the performance, we employ a two-stage Probability Annealing Selection (PAS) framework [38] instead of the traditional teacher-student imitation learning approach, as illustrated in Fig. 3. Further details are in *Appendix B*.

In the first step of training, the policy can access all the information $[\boldsymbol{p}_t, \hat{\boldsymbol{s}}_t, \hat{\boldsymbol{c}}_t, \boldsymbol{g}_t]$ as observation. In addition, we use explicit-implicit dual-state learning similar to [18]. The contact force $\hat{\boldsymbol{c}}_t$ is first encoded by a contact encoder to an implicit latent, and concatenated with explicit privileged state $\hat{\boldsymbol{s}}_t$ to the dual-state $\hat{\boldsymbol{l}}_t = [Enc(\hat{\boldsymbol{c}}_t), \hat{\boldsymbol{s}}_t]$. In addition, we introduce a classification head to guide the policy in learning the connection between the contact force distribution and the trap category. In fact, the category of trap can be seen as a very strong explicit privileged state. We use cross-entropy loss as the classification loss. Previous work [15] uses boolean values as the collision state for prediction. We find this will cause the sim-to-real gap, referring to the experiment Sec. IV-D. Besides, we pre-train the estimator module in the first step using L2 loss to reconstruct $\hat{\boldsymbol{l}}_t$. In summary, the total optimization target is:

$$L_{surro} + L_{value} + L_{recon} + L_{classify} + L_{discriminator}. \tag{7}$$

In the second step of training, the policy can only access $[\boldsymbol{p}_t, \boldsymbol{g}_t]$ as observation. We initialize the weight of the estimator and the low-level RNN copied from the first training step. Probability Annealing Selection is used to gradually adapt policies to inaccurate estimates while reducing the degradation of the Oracle policy performance.

$$\boldsymbol{l}_t = \text{Estimator}(\boldsymbol{p}_t, \boldsymbol{g}_t), \tag{8}$$

$$\boldsymbol{i}_t = \text{Probability Selection}(\boldsymbol{P}_t, \boldsymbol{l}_t, \hat{\boldsymbol{l}}_t), \tag{9}$$

$$\boldsymbol{a}_t = \text{Actor } E_{low}(\boldsymbol{i}_t, \boldsymbol{p}_t, \boldsymbol{g}_t), \tag{10}$$

$$\text{Probability } \boldsymbol{P}_t = \alpha^{iteration}. \tag{11}$$

To enhance training stability, we increase the batch size by using 4,096 parallel robots, each performing 50 steps. Moreover, the episode length is reduced from 20 seconds to 8 seconds. More details of dynamic randomization and trap terrain curriculum are in the *Appendix C* and *Appendix D*.

2) Deployment. Unlike previous works, our policy achieves approximate omnidirectional movement by joystick commands without motion capture or other auxiliary localization techniques. As shown in Fig. 3, the dual-state $\boldsymbol{l}_t$ is predicted by the Estimator Module and Latent Encoder, then passed to the low-level Actor RNN combined with proprioceptive from robot

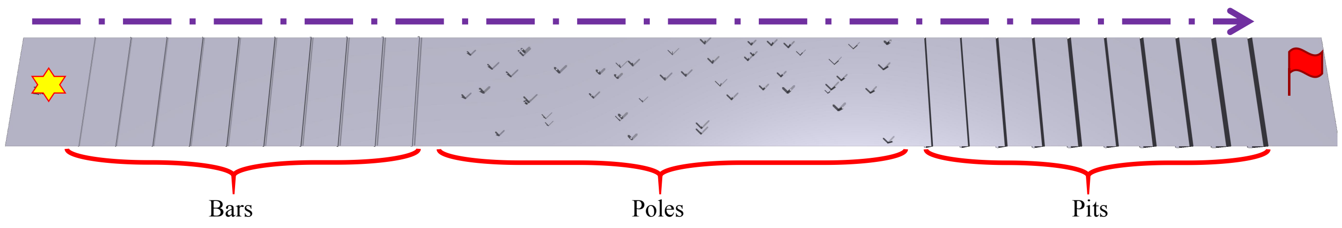

Fig. 5: **Tiny Trap Benchmark.** Robots begin on the left side and must pass through tiny traps to reach the goal on the right side.

sensors. A joystick is used to generate fake goal commands, consisting of constant values for ΔG and Δt.

With the well-designed task reward function and their proportions, the policy learns different movement strategies based on the distance to the goal. For large distance, the robot first rotates in place to face the goal, then moves forward. For medium distance, the robot simultaneously moves toward the goal while turning. For small distance, the robot moves directly to it without turning. This behavior arises from the interplay between $r_{heading}$, which increases when the robot turns toward the goal, and r_{goal}, which decreases with time spent turning. During training, these two rewards reach a dynamic balance, maximizing overall reward. By leveraging this feature, we can achieve different movement patterns using various combinations of goal directions and distances.

When $||\Delta G||_2$ is just above the stop threshold of 0.2, the robot's angular velocity is nearly zero, enabling translational movements such as forward, backward, left, and right. When $||\Delta G||_2 = 1.0$ to the right or left, the robot will continuously turn right or left in place. By leveraging these two tricks, we can accomplish basic operations. Additional combinations of velocities are discussed in Sec. IV-E. Although the robot has not been explicitly trained with constant ΔG and Δt, it can zero-shot deploy with no performance degradation. Furthermore, the robot can perform a variety of movements sequentially by changing ΔG and Δt, without needing to reset the policy or the robot. With standstill state, the robot can start from a stationary state ($||\Delta G||_2 = 0$) and seamlessly execute operations.

In practice, the left side of the joystick controls the direction of the goal command (i.e., the ratio of Δx to Δy), while the right side controls the distance $||\Delta G||_2$ to the goal. Additionally, some keys on the joystick can toggle the value of Δt.

IV. EXPERIMENTS

A. Experiment Setup

We use the IsaacGym [5] for policy training and deploy 4,096 quadruped robot agents on a single NVIDIA RTX 3090. We first train 12,000 iterations on plane terrain, then train 30,000 iterations for both stages on trap terrain. The control policy within both the simulator and the real world operates at a frequency of 50 Hz. We deploy our policy on the Unitree A1 quadruped robot which has an NVIDIA Jetson Xavier NX as the onboard computer. In the real-world deployment, the robot receives the goal command from the joystick through ROS message and runs the low-level control policy to predict desired joint positions for PD control ($K_p = 40, K_d = 0.5$).

B. Tiny Trap Benchmark

We design a new **Tiny Trap Benchmark** in simulation. As shown in Fig. 5, the benchmark consists of a 5m×60m runway with three types of traps evenly distributed along the path. The traps include 10 bars with heights ranging from 0.05m to 0.2m, 50 randomly placed poles, and 10 pits with widths ranging from 0.05m to 0.2m. For each experiment, 1,000 robots are deployed, starting from the left side of the runway and passing through all the traps to reach the right side. We refer to this as the "Mix" benchmark. Additionally, there are separate "Bar," "Pit," and "Pole" benchmarks, each focusing on one specific type of trap, but with triple the number of traps.

The benchmark uses three metrics: **Success Rate**, **Average Pass Time**, and **Average Travel Distance**. A robot is considered successful if it reaches within 0.2m of the target point within 300 seconds, at which point we record its pass time. Failure cases include falling off the runway, getting stuck, or rolling over. For failed cases, the pass time is set to a maximum of 300 seconds. We then calculate the overall success rate and average pass time. At the end of the evaluation, we average the lateral travel distance of all robots. During the evaluation, we use the same fake goal commands as in real-world deployment and guide the robot to stay in the center.

C. Simulation Experiments

We conducted experiments with the following methods:

- Different combinations of joint links. (Prop: Only trained with proprioception).
- Ours w/o goal command: use traditional velocity command, but only train moving forward towards the trap.
- Ours w/ Boolean: without encoder, boolean-value collision states are directly feed into low-level RNN, partly like [15]. Other settings are same as our method.
- RMA [10]: A 1D-CNN serves as an asynchronous adaptation module in the teacher-student training framework.
- MoB [39]: "Learning a single policy that encodes a structured family of locomotion strategies that solve training tasks in different ways."
- HIMLoco [40]: "HIM only explicitly estimates velocity and implicitly simulates the system response as an implicit latent embedding by constrastive learning."

For RMA and HIMLoco, they are designed to deal with common terrains like stairs or uneven planes. We retrain the policy in our trap terrain, but the rewards and training settings have not changed (e.g. randomly sample velocity). This may cause the relatively poor performance of these two methods.

We report the results in Table. I and Table. II. In comparison experiments, our method outperforms others in all metrics. In ablation experiments, policy that utilizes all joint links performs best on "Mix" and perform relatively well on other benchmarks. This proves contact force of every joint is helpful in improving the policy performance.

TABLE I: The comparison results with other locomotion policies in our benchmark.

Method	Success Rate ↑				Average Pass Time (s) ↓				Average Travel Distance (m) ↑			
	Mix	Bar	Pit	Pole	Mix	Bar	Pit	Pole	Mix	Bar	Pit	Pole
Ours	**0.934**	**0.941**	**0.902**	**0.702**	**119.03**	**130.65**	**111.91**	**156.40**	**56.37**	**55.54**	**56.22**	**44.06**
w/o goal command	0.000	0.000	0.000	0.279	300.00	300.00	300.00	254.17	3.41	3.81	11.32	24.15
Ours w/ Boolean	0.779	0.718	0.656	0.658	144.66	169.59	161.17	167.21	52.40	50.12	49.28	41.65
RMA	0.000	0.000	0.000	0.067	300.00	300.00	300.00	291.99	5.27	7.45	10.21	17.17
MoB	0.000	0.000	0.000	0.000	300.00	300.00	300.00	300.00	2.90	4.13	4.42	6.57
HIMLoco	0.000	0.000	0.000	0.000	300.00	300.00	300.00	300.00	3.61	4.16	16.73	16.15

TABLE II: The ablation results of different link. `B`-base link, `H`-hip link, `T`-thigh link, `C`-calf link, and `F`-foot link.

Method	Success Rate ↑				Average Pass Time (s) ↓				Average Travel Distance (m) ↑			
	Mix	Bar	Pit	Pole	Mix	Bar	Pit	Pole	Mix	Bar	Pit	Pole
`BHTCF` (Ours)	**0.934**	**0.941**	**0.902**	0.702	**119.03**	130.65	**111.91**	156.40	**56.37**	**55.54**	**56.22**	44.06
`BTCF`	0.918	0.925	0.901	0.704	122.58	138.98	111.99	158.91	56.21	55.10	55.97	41.59
`TCF`	0.828	0.815	0.820	0.618	144.47	163.72	134.32	176.16	53.03	48.44	52.62	37.77
`TC`	0.904	0.929	**0.902**	0.687	120.88	**125.19**	113.36	159.21	55.58	54.89	54.96	41.54
`BTC`	0.901	0.902	0.899	**0.729**	123.59	136.90	112.15	**153.41**	55.70	53.18	54.17	**46.03**
`T`	0.864	0.919	0.850	0.692	128.97	128.28	122.87	159.60	53.97	53.44	53.23	41.87
Prop `(None)`	0.819	0.874	0.796	0.655	142.84	148.01	139.03	164.72	52.10	50.02	49.85	39.40

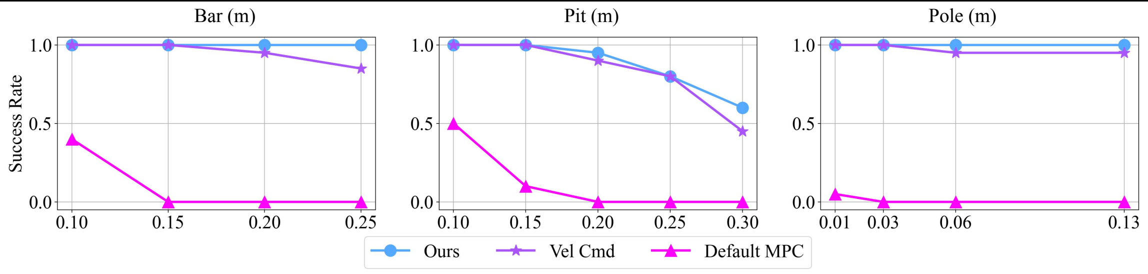

Fig. 6: Real-world experiments. The height of the bar is between the range [0.1m, 0.25m]. The width of the pit is between the range [0.1m, 0.3m]. The width of the pole is between the range [0.01m, 0.13m].

D. *Real-world Experiments*

We deploy the policy on real robots and conduct real-world experiments. The robot achieves zero-shot omnidirectional movement using the joystick commands from Sec. III-D.

The visualized results are shown in Fig. 7. **1) Bar:** The robot learns to step back its front legs after contacting the bar. Even when the hind legs are intentionally trapped, the robot can still detect the collision and lift its hind legs across the bar. In addition, the bar is a little elastic unlike simulations, which also proves the generalization ability of our policy. **2) Pit:** The robot learns to support its body with the other three legs when one leg steps into a void, lifting the dangling leg out of the pit. Additionally, our robot has learned to forcefully kick its legs to climb out when multiple legs are stuck in the pit. **3) Pole:** The robot learns to sidestep to the left or right after colliding with a pole, avoiding the pole by a certain distance before moving forward.

In addition, we deploy and evaluate some other policies for quantitative comparison. For each test, we repeatedly conduct 20 trials and calculate the success rate. The results are in Fig. 6. It shows our method obtains the best performance compared to other baselines. The method with velocity command is not much worse. This is because in the baseline training, we only sample forward velocity command with heading direction guidance rather than traditional random velocity. This greatly increases the performance of the baseline.

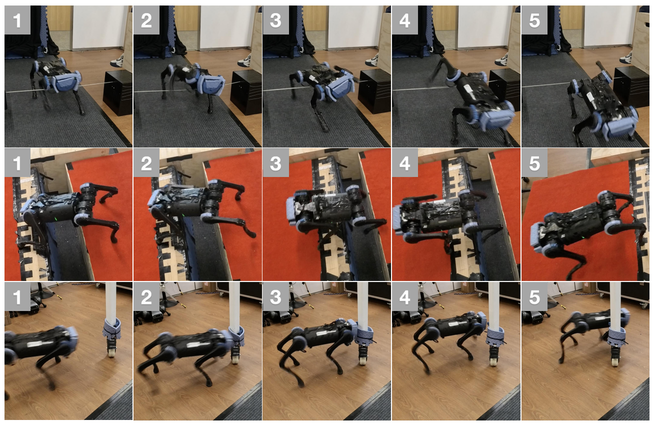

Fig. 7: The quadruped robot passing through tiny traps.

We also show that when a boolean value collision state is directly sent to low-level Actor RNN, the robot will face severe sim-to-real gap in deployment. There are ineliminable gaps between sim and real such as friction and motor damping. When the gap occurs, the latent space will have some noise. The latent space of the boolean value state is very sparse, as shown in Fig. 11; this will cause the policy to misjudge the current state more easily and operate irregularly. As shown in Fig. 9, when the robot moves forward on the plane and suddenly stops for a short time, the gap occurs, and the estimated collision state rises to a large number. If we move backward the robot at that time, the robot will assume it has hit a trap. Since

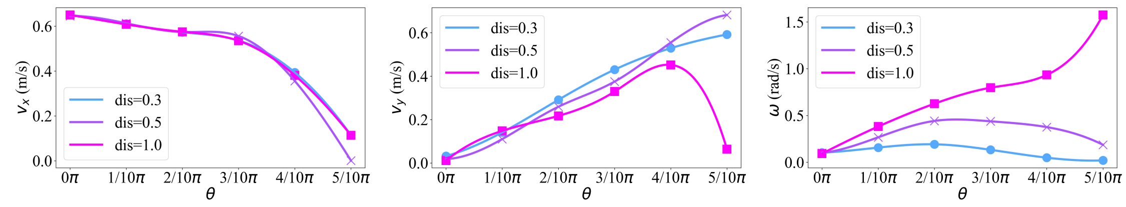

Fig. 8: Experimental results of different ΔG. Different goal distances and directions can cause different movements.

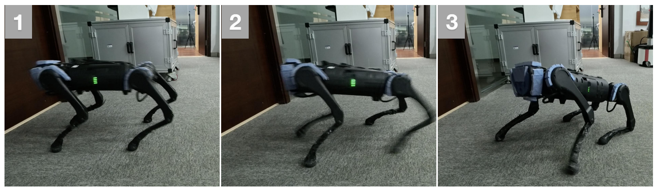

Fig. 9: The robot falls down on the ground with boolean value estimation. While using encoded contact force as estimation value, the robot can continually change the speed with a minimal sim-to-real gap, as shown in appended video.

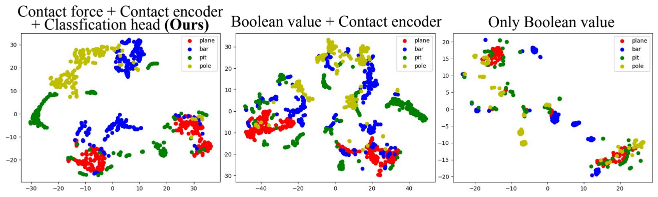

Fig. 11: T-SNE results. Middle: using the boolean collision state and an encoder without classification loss. Right: only using boolean collision without encoder and classification loss.

there is no trap practically, the robot may touch the ground or even fall down. After introducing contact forces and the contact encoder, the sim-to-real gap is mitigated. The robot can better identify traps and operate more stably.

E. Additional Experiments

We conduct additional experiments to further illustrate our method. All experiments in this section are done in simulation.

1) Different Δt of the fake goal command. We tested different Δt values for the fake goal command when passing one 0.2m high bar in Fig. 10. Unlike previous work [3], Δt doesn't indicate policy aggressiveness due to the velocity limit reward. A constant $\Delta t \in [3,5]$ yields the best performance, with fake goal commands performing comparable or better than real ones. Too large or small Δt values degrade performance, likely due to more samples with traps in the middle of the episode.

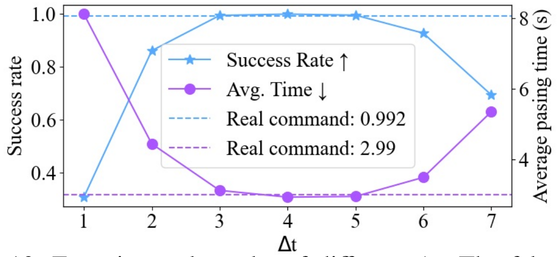

Fig. 10: Experimental results of different Δt. The fake goal command is comparable to or better than the real one.

2) Different ΔG of the fake goal command. As discussed in Sec. III-D, different constant goal commands result in different movements. We recorded V_x, V_y, and ω over one second and averaged the results, as shown in Fig. 8, where $\Delta G = (dis \cdot \cos\theta, dis \cdot \sin\theta, 0)$. The results indicate that: **a.** V_x is approximately proportional to Δx (i. e. $\cos\theta$). **b.** V_y is proportional to θ with small *dis*; V_y remains proportional to small θ but drops quickly as θ approaches $\pi/2$ with large *dis*. **c.** ω is near zero but increases with *dis*. In theory, to execute a command of (V_x, V_y, ω), we can first calculate θ based on the $V_x : V_y$, then select the appropriate *dis* for ω. The limitation of this approach is that $\|V_{x,y}\|_2$ remains constant, allowing control over velocity direction only.

3) T-SNE experiments. We collect the estimated dual-state $\boldsymbol{l}_t$ during the passing time of each trap and visualize them using the t-SNE method [41]. The results are in Fig. 11. It shows that our method with classification head and force contact has a more separable and continuous encoding, which means our policy can identify and react to traps more effectively.

4) Importance analysis: Inspired by saliency map analysis [42], we conduct experiments to analyze the importance of each input in the state space. By calculating the Jacobian matrix and normalizing the results, we find that contact force accounts for about 25% of the importance relative to linear velocity and robot friction. In the contact force space, the base link is the most critical, while the other links show similar importance. These results highlight the crucial role of contact force estimation and the significance of each joint link. Formulas and details can be found in *Appendix F*.

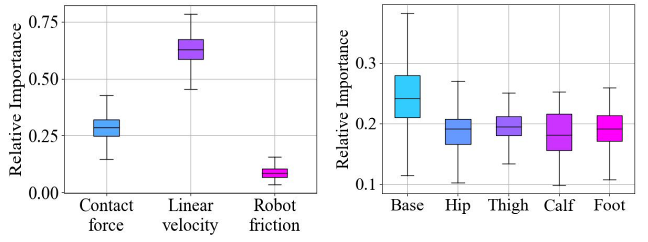

Fig. 12: Importance analysis on latent space and contact links.

V. CONCLUSION

In this work, we propose a novel method for enabling a quadruped robot to pass through tiny traps using only proprioception, avoiding the imprecision of camera images. A contact encoder and classification head are used to better represent contact forces latent. With well-designed rewards and fake goal commands, we enable approximate omnidirectional movement without localization techniques. We deploy our policy on the real robot and demonstrate robustness through extensive experiments.

However, limitations include the inability to control velocity magnitude due to reward settings and struggles with highly deformable traps like rubber bands. Future work could address these by velocity designation and real-world fine-tuning.

APPENDIX

In *Appendix*, we further illustrate the technical details of training, deploying, and experiment setting. *Appendix* mainly has these sections:

- **A. Reward Functions:** the formula of each reward, the definition and analysis of regularization reward, and style reward.
- **B. Network Architectures:** the network architectures of Actor RNN, Critic RNN, Estimator Module, Latent Encoder, and Contact Encoder.
- **C. Dynamic Randomization:** details of the domain randomization and Gaussian noise.
- **D. Trap Terrain Setting in Simulation:** the trap terrain details for training.
- **E. Training Hyperparameters:** the hyperparameters of PPO, contact force, and t-SNE visualization.
- **F. Importance Analysis:** the method and formula for importance analysis.
- **G. Real-world Experiment Settings and Additional Results:** details of the experimental equipment and deployment, and additional experiments in the low-light environment.

A. Reward Functions

The reward function has three components: task reward r_t^T, regularization reward r_t^R, and style reward r_t^S.

In Sec. III-C, we have already introduced the **task reward**, which plays a major role in the training. In addition, **regularization reward** is used to optimize the performance of the robot. "Stall" reward is used to prevent the robot from stopping in the middle of the journey. "Velocity limit" reward is used to slow down the robot and ensure safety. "Joint velocity" reward and "Joint acceleration" reward are used to make the joint movements more stable and smooth. "Angular velocity stability" reward is used to make the base of the robot more stable. "Feet in air" reward is used to improve the gait and prevent the feet from rubbing on the ground. "Balance" reward is used to improve the left-right symmetry. For **style reward**, we first collect a dataset using an MPC controller. The dataset contains a state transition (s_t, s_{t+1}), with a time interval same as the RL policy. $s_t \in \mathbb{R}^{19}$ includes joint positions, base height, base linear velocity, and base angular velocity. We randomly select 200 velocity commands in the simulator. Each command lasts for two seconds and is converted to the next command continually. Following [13], [37], we train a Discriminator $\mathcal{D}_{amp}$ by the following loss function:

$$\begin{aligned} L_{discriminator} = & \mathbb{E}_{(s_t,s_{t+1})\sim \text{MPC}}\left[\left(\mathcal{D}_{amp}(s_t,s_{t+1})-1\right)^2\right] \\ & +\mathbb{E}_{(s_t,s_{t+1})\sim \text{Policy}}\left[\left(\mathcal{D}_{amp}(s_t,s_{t+1})+1\right)^2\right] \\ & +\alpha^{gp}\mathbb{E}_{(s_t,s_{t+1})\sim \text{MPC}}\left[\left\|\nabla \mathcal{D}_{amp}(s_t,s_{t+1})\right\|_2\right], \end{aligned}$$

And then we use $\mathcal{D}_{amp}$ to score the gait performance from policy output (s_t, s_{t+1}):

$$r_{style} = \max\left[0, 1-0.25\left(\mathcal{D}_{amp}(s_t,s_{t+1})-1\right)^2\right]. \quad (12)$$

B. Network Architectures

The details of network architectures are shown in Tab. IV.

TABLE IV: Network architecture details

Network	Type	Input	Hidden layers	Output
Actor RNN	LSTM	$\boldsymbol{p}_t, \boldsymbol{l}_t, \boldsymbol{g}_t$	[512, 256]	$\boldsymbol{a}_t$
Critic RNN	LSTM	$\boldsymbol{p}_t, \hat{\boldsymbol{s}}_t, \hat{\boldsymbol{c}}_t, \boldsymbol{g}_t$	[512, 256]	$\boldsymbol{V}_t$
Estimator Module	LSTM	$\boldsymbol{p}_t, \boldsymbol{g}_t$	[256, 256]	$\boldsymbol{h}_t$
Latent Encoder	MLP	$\boldsymbol{h}_t$	[256, 256]	$\boldsymbol{l}_t$
Contact Encoder	MLP	$\hat{\boldsymbol{c}}_t$	[32, 16]	$\boldsymbol{l}_{t_c}$

TABLE III: Reward functions

Type	Item	Formula	Weight
Task	Get goal	$\frac{1}{0.4+\lVert\Delta G\rVert_2}$	5.0
	Heading	$\begin{cases} \frac{\Delta x}{\lVert\Delta G\rVert_2+\epsilon}, & \lVert\Delta G\rVert_2 \neq 0 \\ 1, & \lVert\Delta G\rVert_2 = 0 \end{cases}$	3.0
	Finish vel	$(\lVert v\rVert + \lVert\omega\rVert)\cdot(\lVert\Delta G\rVert_2 < 0.2)$	-1.0
	Finish pos	$(\sum_{i=1}^{12} \lvert q - q_{default}\rvert)\cdot(\lVert\Delta G\rVert_2 < 0.2)$	-1.0
	Alive	1	3.0
Regularization	Stall	$(\lVert V_{x,y}\rVert_2 < 0.1)\cdot(\lVert\Delta G\rVert_2 > 0.25)$	-2.0
	Vel limit	$(\omega_z < \omega_{limit})\cdot(\lVert v_{x,y}\rVert_2 < v_{limit})$	2.0
	Joint vel	$\lVert\dot{\mathbf{q}}\rVert_2$	-0.002
	Joint acc	$\lVert\ddot{\mathbf{q}}\rVert_2$	-2×10^{-6}
	Ang vel stability	$(\lVert\omega_{t,x}\rVert_2 + \lVert\omega_{t,y}\rVert_2)$	-0.2
	Feet in air	$\sum_{i=0}^{3}(\mathbf{t}_{air,i}-0.3)+10\cdot\min(0.5-\mathbf{t}_{air,i},0)$	0.05
	Balance	$\lVert F_{feet,0}+F_{feet,2}-F_{feet,1}-F_{feet,3}\rVert_2$	-2×10^{-5}
Style	AMP	$\max\left[0,1-0.25\left(\mathcal{D}_{amp}(s_t,s_{t+1})-1\right)^2\right]$	0.1

C. *Dynamic Randomization*

For better sim-to-real transfer, we introduce dynamic randomization, which includes domain randomization and Gaussian noise. We have a series of domain randomization, including base mass, mass position, friction, initial joint positions, initial base velocity, motor strength, and proprioception latency. The random ranges are shown in the Tab. V.

TABLE V: Domain randomization

Parameters	Range	Unit
Base mass	[0, 3]	kg
Mass position of X axis	[-0.2, 0.2]	m
Mass position of Y axis	[-0.1, 0.1]	m
Mass position of Z axis	[-0.05, 0.05]	m
Friction	[0, 2]	-
Initial joint positions	[0.5, 1.5] $\times$ nominal value	rad
Initial base velocity	[-1.0, 1.0] (all directions)	m/s
Motor strength	[0.9, 1.1] $\times$ nominal value	-
Proprioception latency	[0.2, 0.4]	s

Besides, we add Gaussian noise to the input observation, as shown in Tab. VI. This aims to simulate the noise of real robot sensors. Lots of experiments show that with dynamic randomization, the policy can be easily transferred from simulation to the real world without additional training.

TABLE VI: Gaussian noise

Observation	Gaussian Noise Amplitude	Unit
Linear velocity	0.05	m/s
Angular velocity	0.2	rad/s
Gravity	0.05	m/s^2
Joint position	0.01	rad
Joint velocity	1.5	rad/s

D. *Trap Terrain Setting in Simulation*

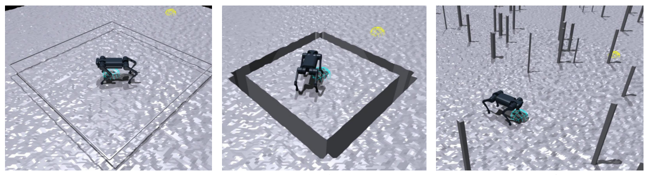

Fig. 13: Trap terrain setting.

We employ “Terrain Curriculum” introduced in previous work [43] for better policy training. Due to the instability of reinforcement learning in early training, it is difficult for the policy to learn the movement in complex traps at once. Therefore, we design a trap curriculum to guide the policy from easy to difficult. The terrain is distributed in 10 rows and 10 columns. The terrains are divided into 4 categories. Each categorey has different traps ranging from easy to difficult. The column numbers of Bar, Pit, Pole, and Plane are 3,2,3,2. To prevent the robot from cheating by detouring, we put the bar and pit in a circle. As shown in Fig. 13, the robot is born inside the circle (blue point) and needs to reach outside the circle (yellow point). The height of the bar increases evenly from 0.05m to 0.25m, with a width randomizing in the range [0.025m, 0.1m]. The width of the pit increases evenly from 0.05m to 0.30m. The number of the pole increases evenly from 10 to 60, with a width randomizing in the range [0.025m, 0.1m]. In addition, we add perlin noise to all of the terrains with an amplitude in the range [0.05m, 0.15m].

E. *Training Hyperparameters*

In our work, we conduct a Policy Optimization algorithm (PPO) as our reinforcement learning method. The hyperparameters are shown in Tab. VII.

TABLE VII: PPO hyperparameters

Hyperparameter	Value
clip min std	0.05
clip param	0.2
desired kl	0.01
entropy coef	0.01
gamma	0.99
lam	0.95
learning rate	0.001
max grad norm	1
num mini batch	4
num steps per env	24

In the training step, we clip the contact force to $[0N, 100N]$.

In addition, we conduct t-SNE visualization in our additional experiments. The hyperparameters are shown in Tab. VIII.

TABLE VIII: T-SNE hyperparameters

Hyperparameter	Value
init	‘random’
perplexity	30
learning rate	200

F. *Importance Analysis*

Assume the input $I \in \mathbb{R}^N$ and the output (action) $O \in \mathbb{R}^M$.

$$O = \text{Policy}(I), \tag{13}$$

First, we obtain the Jacobian matrix $J \in \mathbb{R}^{M \times N}$ by calculating the partial derivative.

$$J = \begin{bmatrix} \frac{\partial O_1}{\partial I_1} & \frac{\partial O_1}{\partial I_2} & \cdots & \frac{\partial O_1}{\partial I_n} \\ \frac{\partial O_2}{\partial I_1} & \frac{\partial O_2}{\partial I_2} & \cdots & \frac{\partial O_2}{\partial I_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial O_m}{\partial I_1} & \frac{\partial O_m}{\partial I_2} & \cdots & \frac{\partial O_m}{\partial I_n} \end{bmatrix}, \tag{14}$$

We take the absolute value for each term of the matrix.

$$J_{abs} = |J|, \tag{15}$$

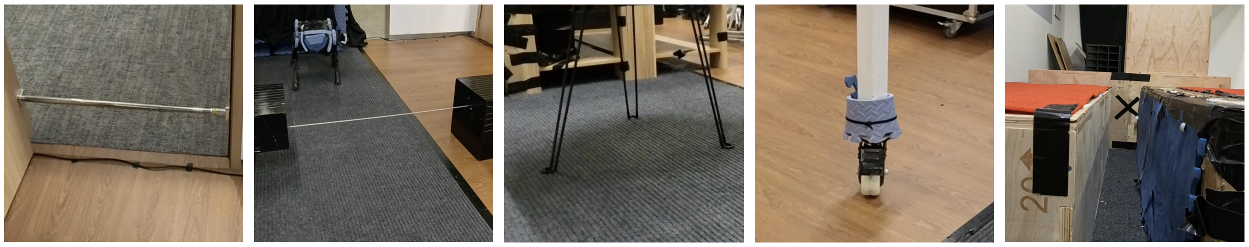
Fig. 14: Real-world traps: Thick *Bar*, Thin *Bar*, Thin *Pole*, Thick *Pole*, *Pit*.

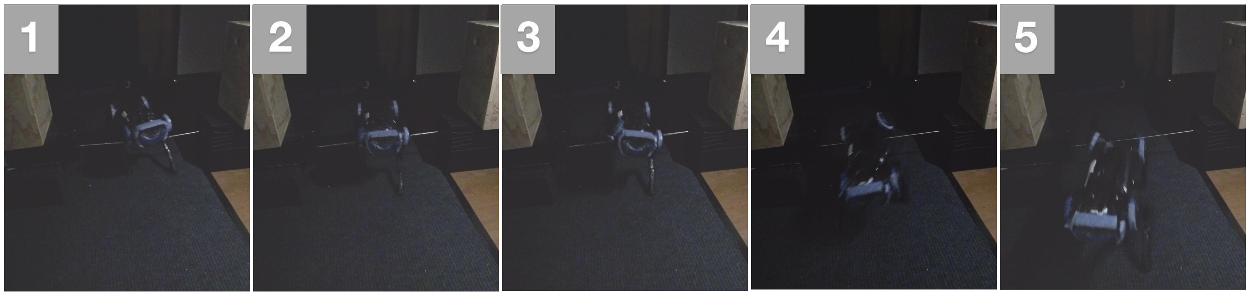

Fig. 15: Crossing *Bar* in low-light environment.

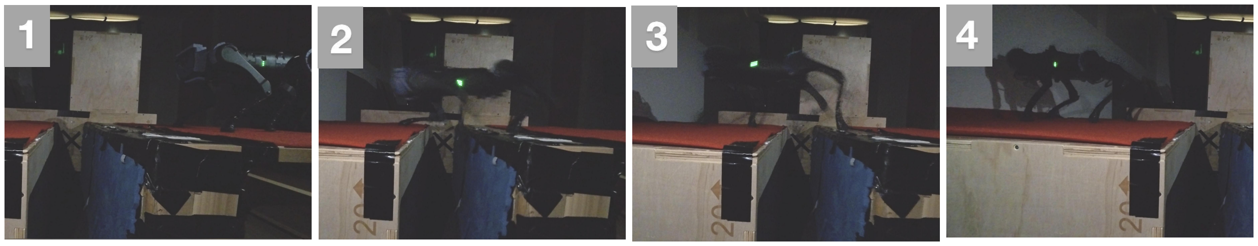

Fig. 16: Crossing *Pit* in low-light environment.

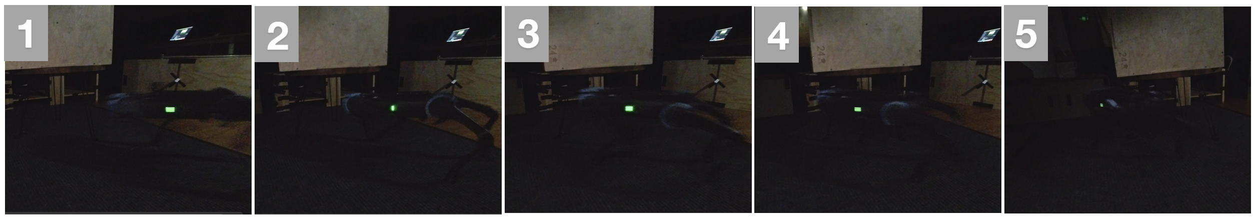

Fig. 17: Crossing *Pole* in low-light environment.

For each input I_i, we sum the corresponding output dimensions and align them with the upper bound U_i and lower bounds L_i to get the importance vector $S \in \mathbb{R}^N$.

$$S_i = (U_i - L_i) \cdot \sum_{j=1}^{M} J_{abs}(j,i), \quad i = 1, 2, \cdots, N \tag{16}$$

For a group of input $\mathcal{G} \in I$, such as Contact force, Linear velocity, etc, we average importance for every input in $\mathcal{G}$.

$$S_{\mathcal{G}} = \frac{\sum_{I_i \in \mathcal{G}} S_i}{\text{num}(I_i \in \mathcal{G})}, \tag{17}$$

The group $\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_k$ is for comparison, we normalize them to get relative importance $R \in \mathbb{R}^k$ for each group.

$$R_i = \frac{S_{\mathcal{G}_i}}{\sum_{j=1}^{N} S_{\mathcal{G}_j}}, \quad i = 1, 2, \cdots, k \tag{18}$$

G. Real-world Experiment Settings and Additional Results

We use common easily accessible items as traps for our real-world experiments, as shown in Fig. 14. For the *Bar* trap, there are two variations: thin bars and thick bars. The thin bars have a diameter of 6mm, while the thick bars measure 20mm in diameter. For the *Pit* trap, we separate two wooden boxes with a height of 40mm by some distance. For the *Pole* trap, there are also thin and thick poles. The thin poles are the legs of a iron table with a diameter of 8mm, while the thick poles include a range of obstacles made from thick iron poles and sticks of varying diameters.

We also conduct experiments in low-light environment, as shown in Fig. 15, Fig. 16, and Fig. 17. The robot can robustly move through different traps even if there is little light. The results demonstrate the effectiveness and importance of proprioception locomotion in scenarios where there is no visual input, such as during nighttime.

References

[1] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on robot learning*. PMLR, 2023, pp. 403–415.

[2] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," *arXiv preprint arXiv:2309.05665*, 2023.

[3] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2497–2503.

[4] C. Zhang, N. Rudin, D. Hoeller, and M. Hutter, "Learning agile locomotion on risky terrains," *arXiv preprint arXiv:2311.10484*, 2023.

[5] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[6] T. Horvat, K. Melo, and A. J. Ijspeert, "Model predictive control based framework for com control of a quadruped robot," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3372–3378.

[7] 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, no. 26, p. eaau5872, 2019.

[8] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, 2022.

[9] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 572–587, 2024.

[10] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[11] Z. Su, X. Huang, D. Ordoñez-Apraez, Y. Li, Z. Li, Q. Liao, G. Turrisi, M. Pontil, C. Semini, Y. Wu, *et al.*, "Leveraging symmetry in rl-based legged locomotion control," *arXiv preprint arXiv:2403.17320*, 2024.

[12] L. Smith, J. C. Kew, T. Li, L. Luu, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Learning and adapting agile locomotion skills by transferring experience," *arXiv preprint arXiv:2304.09834*, 2023.

[13] J. Wu, G. Xin, C. Qi, and Y. Xue, "Learning robust and agile legged locomotion using adversarial motion priors," *IEEE Robotics and Automation Letters*, 2023.

[14] 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.

[15] Y. Cheng, H. Liu, G. Pan, L. Ye, H. Liu, and B. Liang, "Quadruped robot traversing 3d complex environments with limited perception," *arXiv preprint arXiv:2404.18225*, 2024.

[16] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 11 443–11 450.

[17] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[18] S. Luo, S. Li, R. Yu, Z. Wang, J. Wu, and Q. Zhu, "Pie: Parkour with implicit-explicit learning framework for legged robots," *arXiv preprint arXiv:2408.13740*, 2024.

[19] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.

[20] E. A. Sisbot and R. Alami, "A human-aware manipulation planner," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1045–1057, 2012.

[21] D. M. Ebert and D. D. Henrich, "Safe human-robot-cooperation: Image-based collision detection for industrial robots," in *IEEE/RSJ international conference on intelligent robots and systems*, vol. 2. IEEE, 2002, pp. 1826–1831.

[22] S. Takakura, T. Murakami, and K. Ohnishi, "An approach to collision detection and recovery motion in industrial robot," in *15th Annual conference of IEEE industrial electronics society*. IEEE, 1989, pp. 421–426.

[23] S. Morinaga and K. Kosuge, "Collision detection system for manipulator based on adaptive impedance control law," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 1. IEEE, 2003, pp. 1080–1085.

[24] J. Vorndamme, M. Schappler, and S. Haddadin, "Collision detection, isolation and identification for humanoids," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4754–4761.

[25] H. Kuntze, C. W. Frey, K. Giesen, and G. Milighetti, "Fault tolerant supervisory control of human interactive robots," in *IFAC Workshop on Advanced Control and Diagnosis, Duisburg, D*, 2003.

[26] S. Golz, C. Osendorfer, and S. Haddadin, "Using tactile sensation for learning contact knowledge: Discriminate collision from physical interaction," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3788–3794.

[27] S. Haddadin, A. Albu-Schaffer, and G. Hirzinger, "The role of the robot mass and velocity in physical human-robot interaction-part i: Non-constrained blunt impacts," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1331–1338.

[28] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3356–3363.

[29] S. Parusel, S. Haddadin, and A. Albu-Schäffer, "Modular state-based behavior control for safe human-robot interaction: A lightweight control architecture for a lightweight robot," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4298–4305.

[30] J. Van Dam, A. Tulbure, M. V. Minniti, F. Abi-Farraj, and M. Hutter, "Collision detection and identification for a legged manipulator," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 602–13 609.

[31] J. Hwangbo, C. D. Bellicoso, P. Fankhauser, and M. Hutter, "Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3872–3878.

[32] M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, and C. Semini, "Probabilistic contact estimation and impact detection for state estimation of quadruped robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1023–1030, 2017.

[33] M. Maravgakis, D.-E. Argiropoulos, S. Piperakis, and P. Trahanias, "Probabilistic contact state estimation for legged robots using inertial information," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 163–12 169.

[34] V. Barasuol, G. Fink, M. Focchi, D. Caldwell, and C. Semini, "On the detection and localization of shin collisions and reactive actions in quadruped robots," in *International Conference on Climbing and Walking Robots*, vol. 49, 2019, p. 51.

[35] I. M. A. Nahrendra, B. Yu, and H. Myung, "Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5078–5084.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[37] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–20, 2021.

[38] S. Zhu, D. Li, Y. Liu, N. Xu, and H. Zhao, "Cross anything: General quadruped robot navigation through complex terrains," *arXiv preprint arXiv:2407.16412*, 2024.

[39] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Conference on Robot Learning*. PMLR, 2023, pp. 22–31.

[40] J. Long, Z. Wang, Q. Li, L. Cao, J. Gao, and J. Pang, "Hybrid internal model: Learning agile legged locomotion with simulated robot response," in *The Twelfth International Conference on Learning Representations*, 2024.

[41] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[42] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[43] N. Heess, D. Tb, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.