2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

DreamWaQ: Learning Robust Quadrupedal Locomotion With Implicit Terrain Imagination via Deep Reinforcement Learning

I Made Aswin Nahrendra[1], Byeongho Yu[1], and Hyun Myung[1*], *Senior Member, IEEE*

***Abstract*— Quadrupedal robots resemble the physical ability of legged animals to walk through unstructured terrains. However, designing a controller for quadrupedal robots poses a significant challenge due to their functional complexity and requires adaptation to various terrains. Recently, deep reinforcement learning, inspired by how legged animals learn to walk from their experiences, has been utilized to synthesize natural quadrupedal locomotion. However, state-of-the-art methods strongly depend on a complex and reliable sensing framework. Furthermore, prior works that rely only on proprioception have shown a limited demonstration for overcoming challenging terrains, especially for a long distance. This work proposes a novel quadrupedal locomotion learning framework that allows quadrupedal robots to walk through challenging terrains, even with limited sensing modalities. The proposed framework was validated in real-world outdoor environments with varying conditions within a single run for a long distance.**

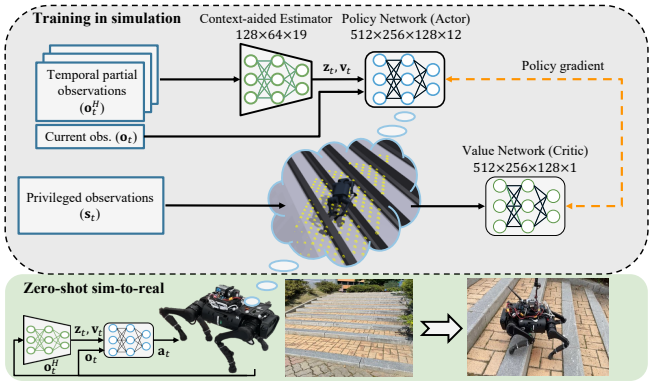

Fig. 1: Overview of DreamWaQ. By learning a locomotion policy in a simulation, the robot can walk through challenging terrains such as stairs with zero-shot sim-to-real.

I. Introduction

In recent years, quadrupedal robots have played an important role in various applications, such as industrial inspection and exploration [1]–[6]. Unlike wheeled mobile robots, quadrupedal robots can traverse unstructured terrains but are relatively difficult to control. Conventional model-based controllers often require a complex pipeline consisting of state estimation, trajectory optimization, gait optimization, and actuator control [1]–[3], [7]–[11]. Such a complex model-based pipeline requires considerable human effort for accurate modeling and rigorous parameter tuning. Moreover, the linearized quadrupedal model often limits its performance, hindering its full capability.

Legged animals can efficiently plan their gait by visually perceiving the surrounding terrains. This natural mechanism has inspired many works on training a perceptive locomotion policy via deep reinforcement learning (RL) that can enable a quadrupedal robot to traverse unstructured terrains [12]–[15]. In these frontier works, the robot is equipped with exteroceptive sensors such as a camera or LiDAR to observe its surroundings. Subsequently, exteroception is used with the controller to plan the robot's trajectory and gait to traverse through the environment safely.

However, exteroception may not always be dependable. Cameras can malfunction in adverse weather and lighting conditions, and while a 3D LiDAR can be utilized to distinguish ground and traversable regions, accurately estimating the physical characteristics of the terrain remains challenging [16]–[18]. For instance, snow may appear as a solid and passable surface, but it is actually soft and pliable. Additionally, tall grass that appears impassable to a camera can still be easily traversed by legged robots.

Meanwhile, proprioceptive sensors, such as an inertial measurement unit (IMU) and joint encoder, are relatively light and robust compared to exteroceptive sensors. Recent works have shown that by combining different proprioception modalities, a quadrupedal robot can learn to estimate its surrounding terrain [19]–[23] and body state [24]. However, these works have a limited empirical demonstration for a long-distance operation with various challenging terrains, where legged robots may fail due to high uncertainties and estimation errors.

Estimating the surrounding terrain's properties via proprioception while learning a locomotion policy requires an iterative process [19], [20], [23]. The policy needs to understand the terrain properties to learn robust behavior. However, to adequately learn the terrain properties, the robot should be able to walk accordingly and explore a wide spectrum of terrain properties. This dilemma is often called the representation learning bottleneck [25], which can hinder optimal policy learning. Therefore, a learning framework that jointly learn a robust policy with an accurate environment representation is required.

In this paper, we proposed a framework called ***Dream Walking for Quadrupedal Robots*** (***DreamWaQ***), that trains a robust locomotion policy for quadrupedal robots with

This work was supported by Korea Evaluation Institute of Industrial Technology (KEIT) grant funded by the Korea Government (MOTIE) (No. 20018216, "Development of Mobile Intelligence SW for Autonomous Navigation of Legged Robots in Dynamic and Atypical Environments for Real Application"). The students are supported by BK21 FOUR.

[1]The authors are with the School of Electrical Engineering at Korea Advanced Institute of Science and Technology (KAIST), Daejeon, 34141, Republic of Korea. {anahrendra, bhyu, hmyung}@kaist.ac.kr
*Corresponding Author: Hyun Myung

only proprioception via a deep RL algorithm. DreamWaQ trains a locomotion policy to implicitly infer the terrain properties, such as height map, friction, restitution, and obstacles. Consequently, the robot can adapt its gait to walk safely through various terrains. We deployed DreamWaQ on a Unitree A1 [26] robot to robustly walk through challenging natural and man-made environments.

In summary, the contributions of this work are threefold:

1) A novel locomotion learning framework via an asymmetric actor-critic architecture is proposed to implicitly imagine terrain properties using only proprioception.
2) A context-aided estimator network is proposed to estimate body state and environmental context jointly. Together with the policy, our method outperforms existing learning-based methods.
3) A robustness and durability evaluation of the learned policy in the real world was conducted through walking in diverse outdoor environments. To the best of our knowledge, this is the first time a Unitree A1, which is significantly smaller that an ANYmal robot, has been demonstrated to sustainably walk on challenging terrain such as hills and yards.[1]

The remainder of this paper is organized as follows. Section II discusses our proposed method thoroughly. Section III presents the experimental setting, results, and an in-depth comparative analysis of the proposed and baseline methods. Finally, Section IV concludes this work and briefly discusses directions for future work.

II. DreamWaQ

A. Preliminaries

In this work, the environment is modeled as an infinite-horizon partially observable Markov decision process (POMDP), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, d_0, p, r, \gamma)$. The full state, partial observation, and action are continuous, and defined by $\mathbf{s} \in \mathcal{S}$, $\mathbf{o} \in \mathcal{O}$, and $\mathbf{a} \in \mathcal{A}$, respectively. The environment starts with an initial state distribution, $d_0(\mathbf{s}_0)$; progresses with a state transition probability $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$; and each transition is rewarded with a reward function, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$. The discount factor is defined by $\gamma \in [0, 1)$. Additionally, in this paper, we define a temporal observation at time t over the past H measurements as $\mathbf{o}_t^H = \begin{bmatrix}\mathbf{o}_t & \mathbf{o}_{t-1} \dots \mathbf{o}_{t-H}\end{bmatrix}^T$. We also define a context vector, $\mathbf{z}_t$, which contains a latent representation of the world state. The context vector is inferred using the method that will be discussed in Section II-C.

B. Implicit Terrain Imagination

Recent works have leveraged the teacher-student training paradigm [27]. Although it has been empirically shown that the student policy is as good as the teacher's, behavior cloning (BC) bounds the student policy's performance with the teacher policy [19], [20], [23]. Moreover, sequentially training the teacher and student networks is data inefficient [24]. The student policy might be unable to explore failure states in which the teacher policy has learned in the early stage of learning using RL. This limitation is because, during BC, the student policy is only provided with good action supervision from the teacher policy.

For learning implicit terrain imagination, we adopted an asymmetric actor-critic architecture [28]. We discovered that the interplay between the policy and value networks in actor-critic algorithms is sufficient for learning a robust locomotion policy that could implicitly imagine the privileged observations, given partial temporal observations. In DreamWaQ, the policy (actor) receives temporal partial observations, $\mathbf{o}_t^H$, as the input, while the value network (critic) receives the full state, $\mathbf{s}_t$, as shown in Fig. 1. In this work, we use $H = 5$. Consequently, the data efficiency during training is significantly increased because only one training phase is required. Moreover, the policy can explore all possible trajectories during training, increasing its robustness through generalization. In this work, the policy is optimized using the proximal policy optimization (PPO) algorithm [29].

1) Policy Network: The policy, $\pi_\phi(\mathbf{a}_t|\mathbf{o}_t, \mathbf{v}_t, \mathbf{z}_t)$ is a neural network parameterized by ϕ that infers an action $\mathbf{a}_t$, given a proprioceptive observation $\mathbf{o}_t$, body velocity $\mathbf{v}_t$, and latent state $\mathbf{z}_t$. $\mathbf{o}_t$ is measured directly from joint encoders and IMU, while $\mathbf{v}_t$ and $\mathbf{z}_t$ are estimated by a context-aided estimator network (CENet), which will be discussed in Section II-C. $\mathbf{o}_t$ is an $n \times 1$ vector defined as follows:

$$\mathbf{o}_t = \begin{bmatrix}\boldsymbol{\omega}_t & \mathbf{g}_t & \mathbf{c}_t & \boldsymbol{\theta}_t & \dot{\boldsymbol{\theta}}_t & \mathbf{a}_{t-1}\end{bmatrix}^T, \tag{1}$$

where $\boldsymbol{\omega}_t$, $\mathbf{g}_t$, $\mathbf{c}_t$, $\boldsymbol{\theta}_t$, $\dot{\boldsymbol{\theta}}_t$, and $\mathbf{a}_{t-1}$ are the body angular velocity, gravity vector in the body frame, body velocity command, joint angle, joint angular velocity, and previous action, respectively.

2) Value Network: The value network is trained to output an estimation of the state value, $V(\mathbf{s}_t)$. Unlike the policy, the value network receives the privileged observation, $\mathbf{s}_t$, which is defined as

$$\mathbf{s}_t = \begin{bmatrix}\mathbf{o}_t & \mathbf{v}_t & \mathbf{d}_t & \mathbf{h}_t\end{bmatrix}^T, \tag{2}$$

where $\mathbf{d}_t$ is the disturbance force applied randomly on the robot's body and $\mathbf{h}_t$ is the height map scan of the robot's surroundings as an exteroceptive cue for the value network. In the proposed DreamWaQ, the policy network is trained to implicitly infer $\mathbf{d}_t$ and $\mathbf{h}_t$ from proprioception.

3) Action Space: The action space is a 12×1 vector, $\mathbf{a}_t$, corresponding to the desired joint angle of the robot. To facilitate learning, we train the policy to infer the desired joint angle around the robot's stand still pose, $\boldsymbol{\theta}_{\text{stand}}$. Hence, the robot's desired joint angle is defined as

$$\boldsymbol{\theta}_{\text{des}} = \boldsymbol{\theta}_{\text{stand}} + \mathbf{a}_t. \tag{3}$$

The desired joint angles are tracked using a proportional–derivative (PD) controller for each joint.

4) Reward Function: Our reward function closely follows other works [12], [19], [20], [22], [24], [30] to highlight the effect of DreamWaQ's components instead of reward tuning. The reward function consists of task rewards for tracking the

[1]Project site: `https://sites.google.com/view/dreamwaq`

TABLE I: Reward function elements. $\exp(\cdot)$ and $\mathrm{var}(\cdot)$ are exponential and variance operators, respectively. $(\cdot)^{\mathbf{des}}$ and $(\cdot)^{\mathbf{cmd}}$ indicate the desired and commanded values, respectively. x, y, and z are defined on the robot's body frame, with x and z pointing forward and upward, respectively. $\mathbf{g}$, $\mathbf{v}_{xy}$, ω_{yaw}, h, $p_{f,z,k}$, $v_{f,xy,k}$, and $\boldsymbol{\tau}$ are the gravity vector projected into the robot's body frame, linear velocities in the xy plane, yaw rate, body height w.r.t. the ground, foot height, foot lateral velocity, and joint torque, respectively.

Reward	Equation (r_i)	Weight (w_i)
Lin. velocity tracking	$\exp\{-4(\mathbf{v}_{xy}^{\mathrm{cmd}} - \mathbf{v}_{xy})^2\}$	1.0
Ang. velocity tracking	$\exp\{-4(\omega_{\mathrm{yaw}}^{\mathrm{cmd}} - \omega_{\mathrm{yaw}})^2\}$	0.5
Linear velocity (z)	v_z^2	-2.0
Angular velocity (xy)	$\boldsymbol{\omega}_{xy}^2$	-0.05
Orientation	$\lvert\mathbf{g}\rvert^2$	-0.2
Joint accelerations	$\ddot{\boldsymbol{\theta}}^2$	-2.5×10^{-7}
Joint power	$\lvert\boldsymbol{\tau}\rvert\lvert\dot{\boldsymbol{\theta}}\rvert$	-2×10^{-5}
Body height	$(h^{\mathrm{des}} - h)^2$	-1.0
Foot clearance	$(p_{f,z,k}^{\mathrm{des}} - p_{f,z,k})^2 \cdot v_{f,xy,k}$	-0.01
Action rate	$(\mathbf{a}_t - \mathbf{a}_{t-1})^2$	-0.01
Smoothness	$(\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2})^2$	-0.01
Power distribution	$\mathrm{var}(\boldsymbol{\tau} \cdot \dot{\boldsymbol{\theta}})^2$	-10^{-5}

commanded velocity and stability rewards to produce a stable and natural locomotion behavior. The details of the reward function are presented in Table I. The total reward of the policy for taking an action at each state is given as:

$$r_t(\mathbf{s}_t, \mathbf{a}_t) = \sum r_i w_i, \tag{4}$$

where i is the index of each reward, as shown in Table I.

The complex reward function for learning a locomotion policy usually includes a motor power minimization term. However, this reward minimizes the overall power without considering each motor's power usage balance. Consequently, in the long run, some motors might overheat faster than others. Therefore, we introduced a power distribution reward to reduce motor overheating in the real world by penalizing motors' power with high variance over all motors used on the robot.

5) Curriculum Learning: We utilized a game-inspired curriculum [12] to ensure progressive locomotion policy learning over difficult terrains. The terrains consisted of smooth, rough, discretized, and stair terrains with ten levels of inclination within $[0°, 22°]$. Furthermore, we found that utilizing the grid-adaptive curriculum [23] for low-speed locomotion results in a better and more stable turning that prevents foot tripping.

C. Context-Aided Estimator Network

The policy trained using the method described in Section II-B requires $\mathbf{v}_t$ and $\mathbf{z}_t$ as input, which can be estimated from proprioception. Prior works estimate $\mathbf{z}_t$ as the latent variable for understanding terrain properties [20], [21], [23]. Additionally, estimating $\mathbf{v}_t$ using a learned network significantly improves the locomotion policy's robustness [24] by eliminating the accumulated estimation drift.

Motivated by those prior works, we discovered that the interplay between terrain and body state estimates significantly improves body state estimation accuracy. Instead

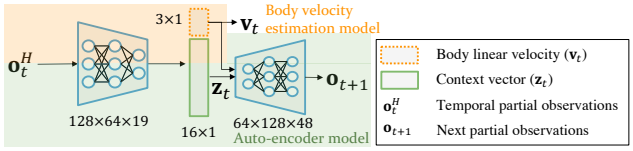

Fig. 2: The architecture of CENet consists of a body velocity estimation model and an auto-encoder model that shares a unified encoder. The shared encoder is trained to provide a robust body state and context estimation jointly.

of only explicitly estimating the robot's state, we propose a context-aided estimator network (CENet) architecture to jointly learn to estimate and infer a latent representation of the environment. The advantages of the proposed CENet are: 1) the network architecture is significantly simplified and runs synchronously during inference owing to the shared encoder architecture; 2) the encoder network can jointly learn the robot's forward and backward dynamics via the auto-encoding mechanism, hence, increasing its accuracy.

CENet consists of a single encoder and a multi-head decoder architecture as shown in Fig. 2. The encoder network encodes $\mathbf{o}_t^H$ into $\mathbf{v}_t$ and $\mathbf{z}_t$. The first head estimates $\mathbf{v}_t$, whereas the second reconstructs $\mathbf{o}_{t+1}$. We leveraged a β-variational auto-encoder (β-VAE) [31]–[33] as the auto-encoder architecture. CENet is optimized using a hybrid loss function, defined as follows:

$$\mathcal{L}_{\mathrm{CE}} = \mathcal{L}_{\mathrm{est}} + \mathcal{L}_{\mathrm{VAE}}, \tag{5}$$

where $\mathcal{L}_{\mathrm{est}}$ and $\mathcal{L}_{\mathrm{VAE}}$ are the body velocity estimation and VAE loss, respectively. For explicit state estimation, we employed a mean-squared-error (MSE) loss between the estimated body velocity, $\tilde{\mathbf{v}}_t$, and the ground truth, $\mathbf{v}_t$, from the simulator as follows:

$$\mathcal{L}_{\mathrm{est}} = MSE(\tilde{\mathbf{v}}_t, \mathbf{v}_t). \tag{6}$$

The VAE network is trained with the standard β-VAE loss, which consists of reconstruction and latent losses. We employed MSE for the reconstruction loss and Kullback-Leibler (KL) divergence [34] as the latent loss. The VAE loss is formulated as

$$\mathcal{L}_{\mathrm{VAE}} = MSE(\tilde{\mathbf{o}}_{t+1}, \mathbf{o}_{t+1}) + \beta D_{\mathrm{KL}}(q(\mathbf{z}_t|\mathbf{o}_t^H) \parallel p(\mathbf{z}_t)), \tag{7}$$

where $\tilde{\mathbf{o}}_{t+1}$ is the reconstructed next observation, $q(\mathbf{z}_t|\mathbf{o}_t^H)$ is the posterior distribution of the $\mathbf{z}_t$, given $\mathbf{o}_t^H$. $p(\mathbf{z}_t)$ is the context's prior distribution parameterized by a Gaussian distribution. We chose a standard normal distribution for the prior distribution because all observations are normalized to have a zero mean and unit variance.

Additionally, bootstrapping from an estimator network during policy network training may increase the sim-to-real robustness of the learned policy [24]. However, we discovered that bootstrapping may also harm the policy's performance because of the large learning noise at the early stage of learning. Therefore, we propose an adaptive bootstrapping (AdaBoot) method that adaptively tunes the bootstrapping probability during training. AdaBoot is controlled by the coefficient of variation (CV), i.e., the ratio of

the standard deviation to the mean, of the episodic reward over m domain-randomized environments. The key idea is that bootstrapping is required when the CV of m agents' rewards is small to make the policy more robust against inaccurate estimation. However, it should not bootstrap when the agents have not learned well enough, as indicated by a large CV in their rewards. We define the bootstrapping probability for each learning iteration as follows:

$$p_{\text{boot}} = 1 - \tanh(CV(\mathbf{R})), \tag{8}$$

where $p_{\text{boot}} \in [0, 1]$ is the bootstrapping probability and $\mathbf{R}$ is an $m \times 1$ vector of episodic rewards from m domain-randomized environments. $CV(\cdot)$, and $\tanh(\cdot)$ are coefficient of variation and hyperbolic tangent operations, respectively. $\tanh$ is used to smoothly upper-bounds $CV(\mathbf{R})$ to one.

III. Experiments

A. Compared Methods

For a comparative evaluation, we compared the following algorithms with access to proprioceptions only:

1) **Baseline** [12]: The policy was trained without any adaptation mechanism.
2) **AdaptationNet** [20], [21]: The policy was trained with an implicit environmental factor encoder using the student-teacher training framework. The policy network consists of 1D convolutional neural network (CNN) layers and multilayer perceptron (MLP) layers.
3) **EstimatorNet** [24]: The policy was concurrently trained with an estimator network that explicitly estimates the body state without a context estimation.
4) **DreamWaQ w/o AdaBoot**: The proposed method without adaptive bootstrapping.
5) **DreamWaQ w/ AdaBoot**: The proposed method with adaptive bootstrapping.

All the methods above were trained using the curriculum strategy and reward functions detailed in Section II. For a fair comparison, we used the same network architecture and fixed the initial random seeds for all methods. All networks used exponential linear units (ELUs) [35] as the activation functions for the hidden layers

B. Simulation

We used the Isaac Gym simulator [36] based on the open-source implementation of [12] to synchronously train the policy, value, and CENet networks for $1{,}000$ iterations. We trained $4{,}096$ agents domain-randomized agents in parallel. The details of the randomized parameters are listed in Table II. For all algorithms, the policy network was trained using PPO with clipping range, generalized advantage estimation factor, and discount factor of 0.2, 0.95, and 0.99, respectively. The networks were optimized using the Adam optimizer [37] with a learning rate of 10^{-3}.

All training was performed on a desktop PC with an Intel Core i7-8700 CPU @ 3.20 GHz, 32 GB RAM, and an NVIDIA RTX 3060Ti GPU. Training using the DreamWaQ algorithm took approximately one hour to generate data equal to approximately 46 days of training in the real world.

TABLE II: Domain randomization ranges applied in the simulation.

Parameter	Randomization range	Unit
Payload	$[-1, 2]$	kg
K_p factor	$[0.9, 1.1]$	Nm/rad
K_d factor	$[0.9, 1.1]$	Nms/rad
Motor strength factor	$[0.9, 1.1]$	Nm
Center of mass shift	$[-50, 50]$	mm
Friction coefficient	$[0.2, 1.25]$	-
System delay	$[0.0, 15.0]$	ms

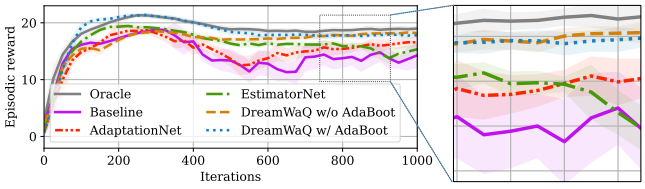

Fig. 3: Learning curves of different algorithms. The results shown are obtained from ten different random seeds. The curves and shaded regions indicate the mean and standard deviation of the reward over ten different seeds, respectively. The oracle policy has access to the height map measurement of the robot's surroundings as in [12].

Fig. 3 compares the learning curves of DreamWaQ against those of all the other methods for learning the locomotion policy of a Unitree A1 robot. It can be seen that even though EstimatorNet initially has a higher mean episodic reward than AdaptationNet, its performance plummets after more iterations because it encounters more difficult terrains after longer training iterations. Conversely, DreamWaQ consistently outperforms all the other methods. Moreover, despite walking without exteroception, DreamWaQ performs almost as well as the oracle policy that has direct access to the surrounding terrain's height map.

C. Real-World Experimental Setup

Real-world experiments were conducted using a Unitree A1 [26] robot. All estimation and control processes were run on an Intel NUC mounted on top of the robot and we used the PyBind interface provided in [38] to send the joint angle command to the robot. An additional onboard PC with a battery added a payload of approximately 500 g to the robot. During inference, the policy runs synchronously with the CENet at 50 Hz. The desired joint angles were tracked using a PD controller with proportional and derivative gains of $K_p = 28$ and $K_d = 0.7$, respectively at 200 Hz.

D. Command Tracking

We evaluated the command tracking performance in a Gazebo simulation to obtain accurate ground truth. The robot was given random commands for ten minutes, and the commands were uniformly sampled from $[-1.0, 1.0]$ every ten seconds. For fair comparison, random commands were generated using the same random seed for each controller. Each controller was run five times with different random seeds to verify repeatability. We measured absolute tracking error (ATE) as the performance metric and constructed a

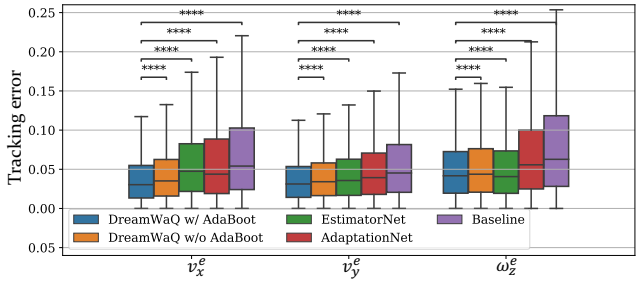

Fig. 4: Command tracking error represented as a boxplot. v_x^e and v_y^e are forward and lateral velocity tracking errors, respectively, measured in m/s. ω_z^e is yaw rate tracking error measured in rad/s. The $****$ annotations indicate measurements with p-value $< 10^{-4}$.

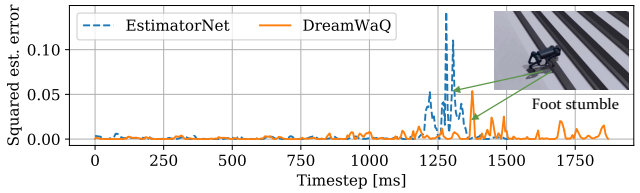

Fig. 5: Estimation error of CENet and EstimatorNet. The superiority of CENet is highlighted when the robot's feet stumbled by stairs.

barplot, as shown in Fig. 4. The significance of the improvement obtained by DreamWaQ against other methods was measured using paired t-test, as shown in Fig. 4, indicating that DreamWaQ consistently outperforms the baselines. Moreover, the proposed AdaBoot method also significantly improved DreamWaQ, owing to its statistical bootstrapping strategy during training.

E. Explicit Estimation Comparison

We simulated the robot walking in a stairs environment to compare the CENet with EstimatorNet in terms of their squared estimation error, as shown in Fig. 5. In the normal walk condition, CENet shows small errors on the flat terrain, thanks to the forward-backward dynamics learning enabled by the VAE's auto-encoding mechanism.

The strength of CENet is highlighted when the robot stumbles down the stairs, where the EstimatorNet fails to estimate the body velocity accurately. In severe cases, inaccurate estimation can lead to catastrophic failure. Conversely, the CENet can accurately estimate the body velocity, enabling the robot to climb the stairs safely. We hypothesize that this is made possible by two factors: 1) the forward-backward dynamics learning provides more accurate estimation in all terrains, and 2) using DreamWaQ, the encoder is jointly trained to predict the terrain properties; hence, it can implicitly reason about the terrain properties, which helps in conditioning the explicit estimation.

F. Robustness Analysis

To test the learned policy's robustness, we perturbed the robot in the simulation with random pushes by applying random velocities with random directions along the x, y, and z axes of the robot's body frame with a one-second interval until it fell. The random push velocities were uniformly sampled from $[-\mathbf{v}_{\text{push}}^{\max}, \mathbf{v}_{\text{push}}^{\max}]$, where $\mathbf{v}_{\text{push}}^{\max} \geq 0$ is the maximum push speed. We also measured the survival rate, i.e., the percentage of the robot's survival time within 30 minutes of a random walk. The result of the robustness test is summarized in Table III.

TABLE III: Robustness test. Bold values indicate results with the most robust performance.

Algorithm	Max. push (m/s)	Survival rate (%)
Baseline	0.511 ± 0.053	20.51 ± 6.44
AdaptationNet	0.714 ± 0.096	82.37 ± 2.49
EstimatorNet	0.871 ± 0.124	80.92 ± 5.73
DreamWaQ w/o AdaBoot	1.015 ± 0.121	90.71 ± 1.25
DreamWaQ w/ AdaBoot	$\mathbf{1.121 \pm 0.164}$	$\mathbf{95.23 \pm 1.61}$

In all methods, the robot mostly fell when there was a significant change in the command vector, requiring the robot to brake and alter its movement quickly. Nevertheless, DreamWaQ is significantly more robust than all the other methods, as quantitatively verified by the high survival rate and maximum push that it can withstand. The robust performance was achieved through the interplay between accurate estimation and robust policy learning of DreamWaQ. Moreover, the proposed AdaBoot method also increases robustness without sacrificing the base performance.

In the real world, DreamWaQ's policy is robust against unstructured terrains. Fig. 6 shows the robot's foot reflex when faced with foot stumbling and slipping. The robot can immediately adapt its gait and stabilize its pose. Owing to the robust and accurate CENet, the robot had no problem in its body velocity estimation and could continue its journey without any performance deterioration.

In Fig. 6(a), the robot exhibits different gaits for going downstairs and upstairs. When going downstairs, the robot tends to tilt its body closer to the ground and maintain its front foot far from the body, which is a key gait pattern for quickly finding a stable foothold. Meanwhile, the robot adapts its gait for going upstairs by significantly increasing its footsteps. This gait is necessary so that the foot can safely overcome the stairs and find a stable foothold while climbing. Moreover, Fig. 6(b) shows the adaptation to slipping, where the robot can immediately detect irregular footholds and adapt its gait pattern. Subsequently, the robot tries to recover its normal pattern and continues to walk.

G. Long-Distance Walk

We deployed the robot on two challenging outdoor courses to demonstrate the robustness of DreamWaQ. Course A was an on-campus yard consisting of many slopes and deformable terrains. Course B was an on-campus hill with an elevation gain of up to 22 m. Courses A and B have a total length of 430 m and 465 m, respectively. The details of the courses are shown in Fig. 7. The robot's trajectory was measured using a real-time kinematic (RTK) GPS [39] with a frequency of 10 Hz, mounted on top of the robot. For complete experiment videos, please refer to the project site[1].

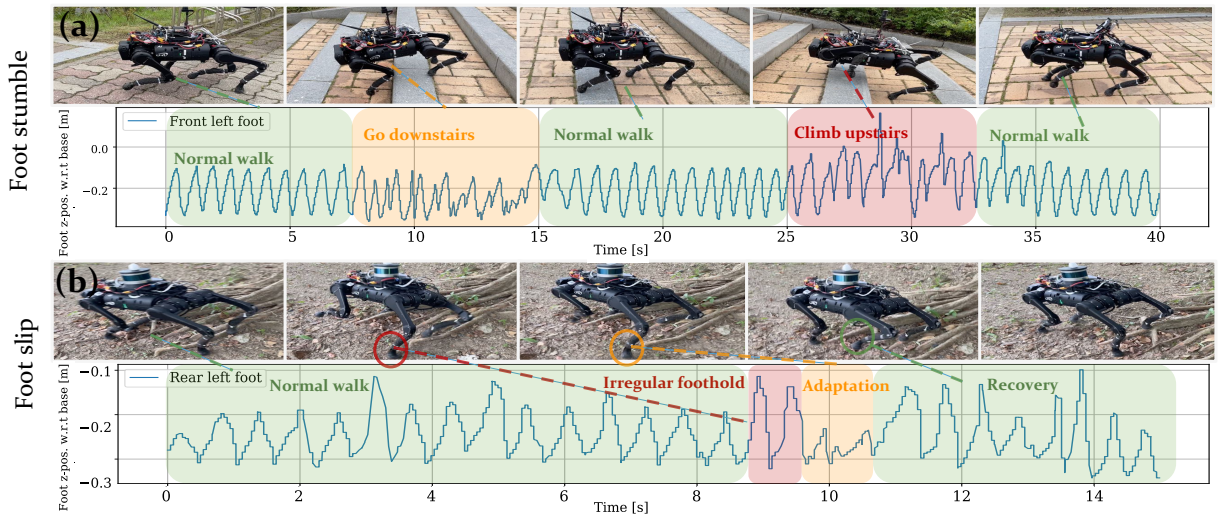

Fig. 6: Foot reflex against uncertainties due to (a) stumbling and (b) slipping in unstructured terrains. Real-time experiment videos are available online[1].

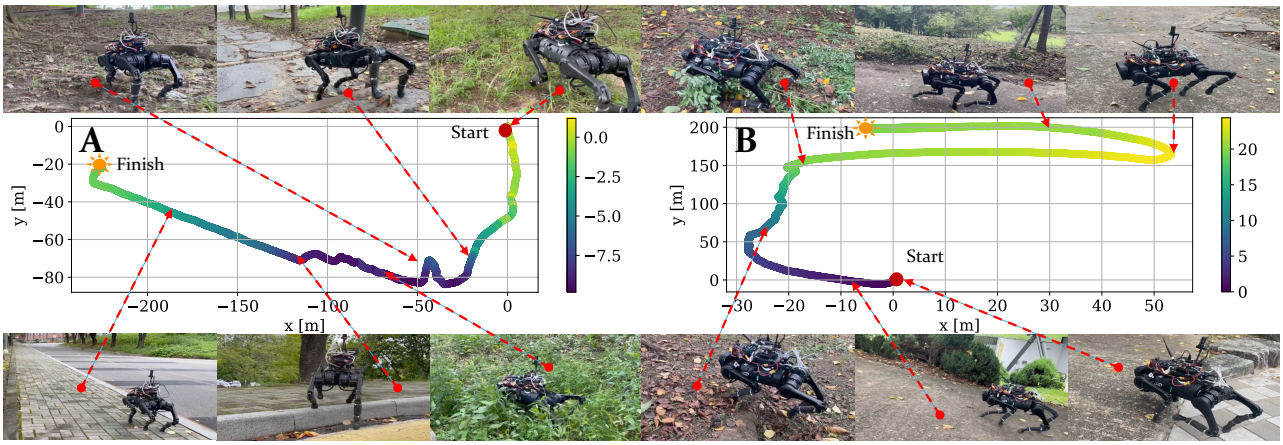

Fig. 7: The outdoor trajectory for testing the performance of the DreamWaQ policy was recorded using an RTK–GPS mounted on the robot. Course A consists of many unstructured natural terrains in yards, while course B is a hiking track. The elevations of both courses relative to the starting point (in [m]) are shown in the color bars.

1) Course A: The robot was challenged in unstructured natural tracks with various slopes in this course. The robot also encountered thick vegetation that trapped the robot's legs. However, the robot successfully adapted its speed by increasing joint power to overcome the trap.

The most challenging part of this course is walking through stairs and deformable slopes. Thanks to the robustness of the policy and accurate estimation of DreamWaQ, the robot could safely walk through the stairs and slopes. We conducted the experiments not only in dry but also in wet terrain conditions after rainfall. While walking down the stairs, the robot faced slippery stairs. Moreover, the robot's feet stepped deeper to the ground on the slopes because of the mud. Nevertheless, our robot, controlled by DreamWaQ, walked through the wet terrain without any difficulties[1].

2) Course B: Course B challenged the robot to climb a moderately high hill. This hiking track consists of man-made asphalt terrain, gravel, and slopes. The experiments were conducted during summer, and the motors heated up quickly. Therefore, we commanded the robot to move slowly to reduce the required torque. Due to the climbing operation, the front legs' motors may easily overheat, and the motor enters the overheat protection mode. Nevertheless, using DreamWaQ, our robot could climb the hill, completing a 465 m trajectory within 10 minutes and reach the hill's summit[1].

IV. Conclusion

In this work, we introduced DreamWaQ, a robust quadrupedal locomotion framework that enables quadrupedal robots to traverse unstructured terrains by relying solely on proprioception. DreamWaQ showed improved performance compared to existing learning-based controllers, and its robustness was demonstrated on a Unitree A1 robot that walked on hills and unstructured yards for approximately ten minutes. DreamWaQ's limitation lies in its adaptation mechanism, where it must first hit the obstacles with its legs. Addressing more complex structures, such as high-rise stairs, is a part of our future work, which requires integrating exteroception into the locomotion system for improved gait planning prior to obstacle contact.

REFERENCES

[1] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "ANYmal – A highly mobile and dynamic quadrupedal robot," in *Proc. IEEE/RSJ international Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 38–44.

[2] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6295–6301.

[3] Y.-H. Shin, S. Hong, S. Woo, J. Choe, H. Son, G. Kim, J.-H. Kim, K. Lee, J. Hwangbo, and H.-W. Park, "Design of KAIST HOUND, a quadruped robot platform for fast and efficient locomotion with mixed-integer nonlinear optimization of a gear train," in *Proc. International Conference on Robotics and Automation (ICRA)*, 2022, pp. 6614–6620.

[4] C. Gehring, P. Fankhauser, L. Isler, R. Diethelm, S. Bachmann, M. Potz, L. Gerstenberg, and M. Hutter, "ANYmal in the field: Solving industrial inspection of an offshore HVDC platform with a quadrupedal robot," in *Field and Service Robotics*, G. Ishigami and K. Yoshida, Eds. Singapore: Springer, 2021, ch. 16, pp. 247–260.

[5] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, *et al.*, "CERBERUS in the DARPA subterranean challenge," *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.

[6] E. M. Lee, D. Seo, J. Jeon, and H. Myung, "QR-SCAN: Traversable region scan for quadruped robot exploration using lightweight precomputed trajectory," in *Proc. 21st International Conference on Control, Automation and Systems (ICCAS)*, 2021, pp. 957–961.

[7] Y. Kim, B. Yu, E. M. Lee, J.-H. Kim, H.-W. Park, and H. Myung, "STEP: State estimator for legged robots using a preintegrated foot velocity factor," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4456–4463, 2022.

[8] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart, "State estimation for legged robots on unstable and slippery terrain," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 6058–6064.

[9] C. Gehring, C. D. Bellicoso, P. Fankhauser, S. Coros, and M. Hutter, "Quadrupedal locomotion using trajectory optimization and hierarchical whole body control," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4788–4794.

[10] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3359–3365.

[11] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, "TAMOLS: Terrain-aware motion optimization for legged systems," *IEEE Transactions on Robotics*, 2022, doi:10.1109/TRO.2022.3186804.

[12] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proc. Conference on Robot Learning (CoRL)*, 2022, pp. 91–100.

[13] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[14] Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, and D. Pathak, "Coupling vision and proprioception for navigation of legged robots," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 17 273–17 283.

[15] W. Yu, D. Jain, A. Escontrela, A. Iscen, P. Xu, E. Coumans, S. Ha, J. Tan, and T. Zhang, "Visual-locomotion: Learning to walk on complex terrains with vision," in *Proc. Conference on Robot Learning (CoRL)*, 2021, pp. 1291–1302.

[16] H. Lim, M. Oh, and H. Myung, "Patchwork: concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3D LiDAR sensor," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6458–6465, 2021.

[17] M. Oh, E. Jung, H. Lim, W. Song, S. Hu, E. M. Lee, J. Park, J. Kim, J. Lee, and H. Myung, "TRAVEL: Traversable ground and above-ground object segmentation using graph representation of 3D LiDAR scans," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7255–7262, 2022.

[18] S. Lee, H. Lim, and H. Myung, "Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3D point cloud," *arXiv:2207.11919*, 2022.

[19] 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.

[20] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," in *Proc. Robotics: Science and Systems*, 2021.

[21] Z. Fu, A. Kumar, J. Malik, and D. Pathak, "Minimizing energy consumption leads to the emergence of gaits in legged robots," in *Proc. Conference on Robot Learning (CoRL)*, 2021, pp. 928–937.

[22] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions," *arXiv:2203.15103*, 2022.

[23] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," in *Proc. Robotics: Science and Systems*, 2022.

[24] 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.

[25] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," in *Proc. International Conference on Learning Representations (ICLR)*, 2021.

[26] "Unitree A1," accessed on 2022.08.24. [Online]. Available: https://m.unitree.com/products/a1

[27] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Proc. Conference on Robot Learning (CoRL)*, 2020, pp. 66–75.

[28] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," in *Proc. Robotics: Science and Systems*, 2018.

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

[30] 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.

[31] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv:1312.6114*, 2013.

[32] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β – VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. International Conference on Learning Representations (ICLR)*, 2017.

[33] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β – VAE," *Advances in Neural Information Processing (NeurIPS) Workshop on Learning Disentangled Representations*, 2017.

[34] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[35] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *Proc. International Conference on Learning Representations (ICLR)*, 2016.

[36] 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," *Advances in Neural Information Processing Systems, Track on Datasets and Benchmarks*, 2021.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[38] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in *Robotics: Science and Systems*, 07 2020.

[39] "H-RTK F9P Helical GPS," accessed on 2022.09.02. [Online]. Available: http://www.holybro.com/product/h-rtk-f9p/