In *In Proceedings of the 2023 IEEE International Conference on Robotics and Automation* (ICRA 2024), Yokohama, Japan, May, 2024

Dexterous Legged Locomotion in Confined 3D Spaces with Reinforcement Learning

Zifan Xu[1], Amir Hossain Raj[2], Xuesu Xiao[2], and Peter Stone[1,3]

***Abstract*— Recent advances of locomotion controllers utilizing deep reinforcement learning (RL) have yielded impressive results in terms of achieving rapid and robust locomotion across challenging terrain, such as rugged rocks, non-rigid ground, and slippery surfaces. However, while these controllers primarily address challenges *underneath* the robot, relatively little research has investigated legged mobility through confined 3D spaces, such as narrow tunnels or irregular voids, which impose *all-around* constraints. The cyclic gait patterns resulted from existing RL-based methods to learn parameterized locomotion skills characterized by motion parameters, such as velocity and body height, may not be adequate to navigate robots through challenging confined 3D spaces, requiring both agile 3D obstacle avoidance and robust legged locomotion. Instead, we propose to learn locomotion skills end-to-end from goal-oriented navigation in confined 3D spaces. To address the inefficiency of tracking distant navigation goals, we introduce a hierarchical locomotion controller that combines a classical planner tasked with planning waypoints to reach a faraway global goal location, and an RL-based policy trained to follow these waypoints by generating low-level motion commands. This approach allows the policy to explore its own locomotion skills within the entire solution space and facilitates smooth transitions between local goals, enabling long-term navigation towards distant goals. In simulation, our hierarchical approach succeeds at navigating through demanding confined 3D environments, outperforming both pure end-to-end learning approaches and parameterized locomotion skills. We further demonstrate the successful real-world deployment of our simulation-trained controller on a real robot.**

I. Introduction

Quadruped robots capitalize on their systems' many Degrees of Freedom (DoFs) and enjoy superior mobility and versatility to locomote through extremely challenging environments, compared to conventional locomotion modalities such as wheeled and tracked systems [1]. By coordinating all the joint angles on their four limbs, quadruped robots can quickly react to unstructured environments, such as rugged rocks, non-rigid ground, and slippery surfaces, and maintain a stable body pose while moving forward.

However, such advanced mobility and versatility comes at a price: in order to coordinate the many DoFs to efficiently maintain torso stability and progress forward, sophisticated planning and control algorithms are necessary to run in real time and react quickly to any environmental constraints and disturbances. Classical gait-based controllers, introduced decades ago [2]–[4], are inspired by the quadruped robots' biological counterparts, e.g., dogs and horses, and are able to move them through moderately challenging terrain. Recently, with the advances in machine learning, robotics researchers have also applied learning methods to enable emergent quadruped locomotion behaviors [5]–[9]. These learning-based methods can often produce better locomotion performance compared to classical hand-crafted approaches, enabling complex locomotion skills [10], [11], safe and agile performance [12], [13], or superior robustness against a variety of unstructured environments [14]–[16].

Fig. 1: Dexterous quadruped legged locomotion in real-world confined 3D spaces.

While current quadruped locomotion techniques allow robots to robustly move through different terrain that give rise to challenges from *underneath* the robot, there is limited research on how a hyper-redundant quadruped robot can move through confined 3D environments which impose *all-around* constraints from 360° surroundings (Fig. 1), e.g., through narrow cave networks [17], complex industrial plants [18], and cluttered indoor environments (e.g., underneath a coffee table) [19], where narrow tunnels or irregular voids are not uncommon. In those spaces, robots not only need to maintain stability due to the underlying terrain and protrusions from the ground, but also precisely maneuver their torso and limbs to squeeze between 3D obstacles from walls and ceilings with a variety of body poses, e.g., large roll and pitch angles with asymmetric, acyclic, and irregular limb movements. In some scenarios, a stable torso may not even be possible to maintain, requiring the robot to lean against obstacles to enable obstacle-aided locomotion [20].

These demanding behaviors are not likely to emerge in transitional parameterized locomotion skills trained by an RL agent tracking motion parameters, such as linear and angular velocities and torso orientations. Instead, we hypothesize that

[1]The University of Texas at Austin zfxu@utexas.edu, pstone@cs.utexas.edu. [2]George Mason University {araj20, xiao}@gmu.edu. [3]Sony AI. This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (FAIN-2019844, NRT-2125858), ONR (N00014-18-2243), ARO (E2061621), Bosch, Lockheed Martin, and UT Austin's Good Systems grand challenge. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

such dexterous locomotion, combined with navigation and manipulation, can only be developed by end-to-end learning of goal-reaching tasks in diverse and confined 3D spaces. In addition, to address the inefficiency of reaching faraway navigation goals, we develop a hierarchical locomotion system, which is composed of a high-level sampling-based planner that plans a feasible body pose as a local goal in front of the robot, and a low-level locomotion policy that is trained with RL to reach this local goal. The sampling-based high-level reactive planner is able to quickly respond to the limited and mostly obstructed robot perception in those confined 3D spaces in a computationally efficient manner to assure real-time execution. The low-level RL policy can robustly move the many limb joints to follow the high-level local poses while considering the all-around constraints not only from the terrain underneath but also 360°-surrounding irregular obstacles. Such a hybrid setup allows the low-level RL policy to fully exploit the hyper-redundant solution space with better sample efficiency compared with pure end-to-end learning using sparse goal-reaching reward. By training on sequences of local goals planned by the high-level controller, the low-level RL policy also enables smooth transitions between local goals, which guides the long-term navigation to distant global goals. The hierarchical training pipeline outperforms both pure end-to-end learning and learning of parameterized locomotion skills. Our main contributions are as follows:

1) We introduce a novel hybrid approach that combines long-term path planning from a classical planner with short-term local goal-reaching via an RL-based locomotion controller.
2) We provide a procedural environment generation pipeline for generating diverse and challenging confined 3D spaces.
3) We investigate the limitations of pre-defined parameterized locomotion skills and pure end-to-end learning in traversing 3D confined spaces.

II. Related Work

In this section, we review related work on both classical and learning-based quadruped locomotion.

A. Classical Quadruped Locomotion

Quadruped locomotion has been a focus of the robotics research community for decades. Starting from building reliable legged hardware [1], [21]–[24], researchers have also developed autonomous navigation and locomotion techniques to move these highly articulated robots in the real world [3], [25], [26]. Most research into quadruped locomotion focus on adapting robot joint angles in the form of gaits [3] to challenging underlying terrain, including rugged rocks, non-rigid ground, and slippery surfaces. With sophisticated gait planners and controllers, quadruped robots can overcome the limitations of conventional wheeled mobile robots and successfully move through a variety of hard-to-reach spaces. However, the hyper redundancy of the many DoFs has led to capabilities beyond just overcoming the terrain challenges from underneath the robot: by carefully coordinating the limb joints, quadruped robots can also fit in confined 3D spaces through a combination of dexterous locomotion, manipulation, and navigation.

One line of research into navigating legged robots through confined 3D spaces is based on whole-body planning-based methods. Mathieu et al. [27] showed that the acyclic gait planner, first introduced by Tonneau et al. [28], can be applied to multilegged robots. Inspired by this work, Buchanan et al. [29] integrated the acyclic gait planners into a hierarchical system that uses a high-level perceptive trajectory planner that plans intermediate body poses for the low-level gait planner to follow. Although the method improved upon Tonneau et al. [28] in a way that takes the precise collision models of the legs into consideration, which enables navigation in very confined 3D spaces, the system still suffers from relatively slow speed caused by frequent replanning and the complicated hierarchical structure. In contrast, our proposed approach is highly efficient, leveraging a sampling-based high-level reactive planner and an RL-based low-level locomotion policy to quickly produce joint angle commands and move the robot through confined 3D spaces.

B. Learning-Based Quadruped Locomotion

Researchers from the robot learning community have also investigated quadruped locomotion using data-driven approaches [5]–[9]. Thanks to the vast amount of simulated trial-and-error experiences gathered using RL, quadruped robots are able to learn superior locomotion skills [10], [11], speed [30], [31], stability [14]–[16], [32], and energy efficiency [11], [13] compared to the classical methods.

However, previous research on RL-based locomotion controllers primarily focuses on learning a limited set of gait parameters such as leg velocity, step frequency, and step amplitude. Such predefined gait patterns might be inefficient when maneuvering robots through complex and confined environments. To overcome these limitations, Rudin et al. [10] proposed to learn a locomotion controller using end-to-end RL training on position-based locomotion tasks. Specifically, the robot is only rewarded positively for reaching a goal position, which allows the agent to select its own path and gait beyond the predefined gaits. However, it is unclear whether acyclic and asymmetrical gait patterns can emerge from such end-to-end learning pipeline in order to react in real-time to obstacles in confined 3D spaces. Moreover, such an end-to-end training can be inefficient when the goal is far away from the robot. In contrast, our proposed approach aims at using RL to efficiently enable the full potential of highly articulated quadruped robots beyond only challenging terrain but also confined 3D spaces that require close and accurate coordination of all the joint angles to fit the robot through 360°-surrounding obstacles (Fig. 1).

III. Approach

In this paper, the objective is to learn a quadruped locomotion controller for goal-reaching tasks in confined 3D spaces with RL. Each task in this context is distinctly characterized by a goal location denoted as $(x_g, y_g) \in \mathbb{R}^2$, representing

a 2D coordinate within the world's reference frame. The locomotion controller is represented by a neural network policy $\pi(a \mid o)$ that takes as input observations o and gives as output joint position commands a. This section introduces three distinct approaches: end-to-end dexterity, hierarchical dexterity, and parameterized motor skills.

A. Observation and Action Spaces

The observation and action spaces are shared across the three approaches except for the definition of commands. Specifically, an observation, denoted as $o = (g, q, \dot{q}, h, c)$, constitutes a five-element tuple, each of which is elaborated as follows:

a) Gravity Vector ($g \in \mathbb{R}^3$): This element characterizes the orientation of the gravity vector within the robot's body frame, and it is acquired through the use of an Inertial Measurement Unit (IMU).

b) Proprioceptive States ($q \in \mathbb{R}^{12}$ and $\dot{q} \in \mathbb{R}^{12}$): These elements contain 12-dimensional joint angles and joint velocities measured by motor encoders.

c) Height Field ($h \in \mathbb{R}^{220}$): This element is measured following a methodology similar to Agarwal et al. [15]. However, it extends beyond the measurement of just the floor height to address terrain challenges. Instead, it captures both floor and ceiling heights to capture all-around obstacles. Precisely, the height field h is represented as two 10×11 matrices, encompassing measurements of floor and ceiling heights. These measurements are taken at predefined 10×11 scandots evenly distributed across a 1-meter by 1-meter area situated in front of the robot. Subsequently, the height field is flattened and concatenated with the other components of the observation.

d) Commands (c): Locomotion controllers operate under specified commands, which are included as a part of the observation. The definition of commands c varies across the three different approaches. For example, the end-to-end dexterity and hierarchical dexterity are commanded with the global and local goal locations respectively. We provide the formal definitions of commands in the subsequent sections.

With regards to actions, each action $a \in \mathbb{R}^{12}$ assigns joint position targets at a frequency of 50 Hz for a Proportional-Derivative (PD) controller. The controller itself operates at a frequency of 200 Hz with the proportional and derivative gains remaining consistent with the values established in prior work [11].

B. End-to-End Dexterity

In the end-to-end dexterity approach, the commands denoted by c are defined as the relative goal position $(x_r, y_r) \in \mathbb{R}^2$, representing the x-y coordinates within the robot's body frame of reference. This means that the policy always takes the relative position of the global goal, and it is trained to directly compute the low-level motor commands necessary for navigating to that global goal. Motivated by a similar end-to-end training pipeline introduced by Rudin et al. [10], the end-to-end dexterity approach incorporates four distinct reward terms: (1) goal-reaching reward; (2) stalling reward; (3) exploration reward; (4) penalty reward. The reward terms are elaborated as follows:

a) Goal-reaching reward:

$$r_g = \begin{cases} +1, & \text{if } \sqrt{x_r^2 + y_r^2} < d_g, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

This reward term assigns +1 when the agent reaching the global goal within a threshold distance denoted as d_g. Then, the episode terminateds.

b) Stalling reward:

$$r_s = \begin{cases} -1, & \text{if } \sqrt{v_x^2 + v_y^2} < 0.1 m/s, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Here v_x and v_y are the robot's linear velocity in the x and y directions in robot's own frame of reference. The stalling reward penalizes the robot for standing still. A negative reward is assigned at every time step when the robot's linear velocity is smaller then 0.1 m/s.

c) Exploration reward: Due to the sparse nature of the main reward, it is beneficial to add a reward term encouraging exploration at the beginning of training. In order to bias the policy to walk towards the global goal, the exploration reward term incentivizes any base velocity in the correct direction. The reward is defined as

$$r_e = \frac{v_x x_r + v_y y_r}{\sqrt{(v_x^2 + v_y^2)(x_r^2 + y_r^2)}}.$$

d) Penalty reward: We penalize joint accelerations, joint torques, collisions, and abrupt actions changes. The penalty reward is defined as follows

$$r_{\text{penalties}} = -c_1 ||\ddot{q}||^2 - c_2 ||\tau||^2 - c_3 N_c - c_4 ||a - a_{-1}||^2.$$

where q, τ, N_c, a, a_{-1} represent joint positions, joint torques, number of collisions, actions, and the actions from last time step respectively, and c_{1-4} are scaling constants. A collision is recorded whenever the thighs, calf, or torso of the body collide with an obstacle or the floor.

C. Hierarchical Dexterity

In contrast to direct navigation to the final goal position, the hierarchical dexterity approach involves a hierarchical system with a classical planner that constantly computes local goals, following which the robot can reach the global goal location, and a locomotion controller that receives commands of local goal locations. Therefore, c is defined as the local goal position $g_l = (x_l, y_l)$ within the robot's body frame of reference.

However, planning in 3D space is very expensive and infeasible during real-world deployment where the entire 3D space is not known a priori, i.e., obstacles in the confined 3D space may obstruct the sensors' fields of view. Therefore, we employ a reactive sampling-based planner, which only uses the available information locally, checks the validity[1]

[1]To avoid expensive whole-body collision checking, the validity of a candidate pose is checked based on collisions between only the torso and the obstacles. Therefore, the planner only provides a rough guidance to the RL locomotion controller.

of a few pre-defined candidate poses, and selects the valid candidate pose that is closest to the final goal position. These candidate poses are expressed as 6-DoF states $p = (x, y, h_z, \phi, \theta, \psi)$, where (x, y) represents a 2D coordinate in the body-frame, h_z denotes the body height, and (ϕ, θ, ψ) denote roll, pitch, and yaw angles, respectively. Then, the x-y coordinate of the selected candidate pose is used as the local goal location (x_l, y_l). Notably, this planning process can be efficiently computed in parallel on a GPU.

The reward design closely matches that of the end-to-end dexterity approach, with the global goal replaced by the local goal in all reward terms.

D. Parameterized Motor Skills

Instead of allowing the robot to learn its own motor skills, the parameterized motor skills approach only searches within the parameter space of a predefined motor skill. This approach operates under the assumption that these predefined motor skills are sufficiently effective for moving the robot through 3D confined spaces, and learning on this smaller solution space is potentially more efficient.

In this approach, the commands c are defined as the parameters of motor skills, specifically denoted as $(v_x^{cmd}, v_y^{cmd}, \omega_z^{cmd}, h_z^{cmd}, \theta^{cmd}, \psi^{cmd})$. Here, v_x^{cmd} and v_y^{cmd} represent the target linear velocities in the body-frame x- and y-axes, while ω_z^{cmd} signifies the target angular velocity around the yaw axis. Additionally, h_z^{cmd}, θ^{cmd}, and ψ^{cmd} correspond to the target height, roll, and pitch angles, respectively.

Notably, unlike prior works that mostly train the policy on randomly sampled commands [11], [33], the parameterized motor skills approach trains the policy on the real command distribution of the task of moving through 3D confined spaces. To achieve this, the approach utilizes the same reactive sampling-based planner employed in the hierarchical dexterity approach. During training, the commands always direct the robot toward the planned local goal poses. Specifically, let $p_l = (x_l, y_l, h_{zl}, \phi_l, \theta_l, \psi_l)$ be the body pose of the planned target local goal. The target linear velocity is a constant 0.5m/s, directed toward the local goal location, which is defined by $v_x^{cmd} = 0.5 \cdot x_l / \sqrt{x_l^2 + y_l^2}$ and $v_y^{cmd} = 0.5 \cdot y_l / \sqrt{x_l^2 + y_l^2}$. ω_z^{cmd} is a constant 1.57 rad/s, maintaining a fixed rate of rotation towards the target yaw angle. h_z^{cmd}, θ^{cmd}, and ψ^{cmd} are kept the same as the target local goal.

To learn these parameterized motor skills, the approach incorporates a *velocity tracking reward* that encourages the robot's linear and angular velocity to be the same as the target velocities, and a *pose tracking reward* that penalizes deviations of the robot's height, roll, and pitch angles from the target orientations. The rewards are formally defined as follows.

a) Velocity Tracking Reward: This reward component motivates the agent to reach the local goal with a constant linear velocity pointing to the local goal location and a constant angular velocity turning the robot to the target yaw angle. The velocity tracking reward contains two reward terms r_{vxy} and r_{wz} defined as

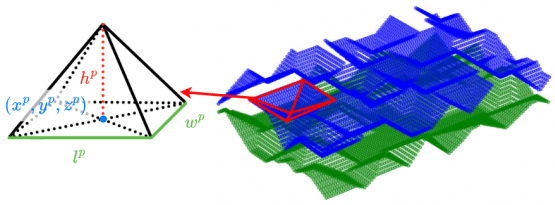

Fig. 2: An example of confined 3D environments constructed from random pyramids (right) and an illustration of pyramid parameters (left).

$$r_{vxy} = \exp\{\frac{-(v_x - v_x^{cmd})^2 - (v_y - v_y^{cmd})^2}{\sigma_{vxy}}\} \quad (3)$$

$$r_{wz} = \exp\{\frac{-(w_z - w_z^{cmd})^2}{\sigma_{wz}}\} \quad (4)$$

Here w_z is the robot's angular velocity around the yaw axis. σ_{vxy} and σ_{wz} are two hyper-parameters.

b) Pose Tracking Reward: This reward component penalizes the robot for not using the target body poses to reach the local goal position. It contains the following three reward terms:

$$r_h = (h_z - h_z^{cmd}), r_\theta = (\theta - \theta^{cmd}), r_\psi = (\psi - \psi^{cmd}).$$

IV. Experiments

This section provides an overview of the training processes for the three approaches specified in Section III. In Section IV-A, we describe the procedural pipeline used for generating confined 3D environments, while Section IV-B delves into the details of training and evaluating the learned locomotion policies for each approach.

A. Environments

The 3D confined environments are constructed from randomly generated $2 \times n_r \times n_c$ matrices of pyramids, positioned on both the ceilings and the floors. As shown in Fig. 2, each pyramid is uniquely characterized by a a set of parameters $(x^p, y^p, z^p, l^p, w^p, h^p)$. Here, x^p, y^p, and z^p represent the 3D coordinate of the center of the base. The base itself is a rectangle with l^p and w^p denoting its length and width, while h^p represents the height of the tip.

The set of parameters of an element indexed by (u, i, j) from the $2 \times n_r \times n_c$ pyramid matrix is denoted as $(x^p_{u,i,j}, y^p_{u,i,j}, z^p_{u,i,j}, l^p_{u,i,j}, w^p_{u,i,j}, h^p_{u,i,j})$, signifying a pyramid located at row i and column j, with u taking values from the set $\{0, 1\}$ to distinguish between pyramids on the floor ($u = 0$) and those on the ceiling ($u = 1$). The range of values for these parameters are specified in Table I. Here, Uniform(a, b) denotes a uniform distribution in the range of $[a, b]$.

The x-y coordinates of the pyramids roughly adhere to a 2D grid with slight variances randomly sampled from the range of $[-0.3, 0.3]$. In the table, Δx and Δy are intervals of the grid in x- and y-axis. The height of the base is set to 0 for pyramids on the floor and 0.5 meters for pyramids on the ceiling. The length and width of the base are randomly

Parameter	Values
$x^p_{u,i,j}$	$\Delta x \cdot i + \text{Uniform}(-0.3, 0.3)$
$y^p_{u,i,j}$	$\Delta y \cdot j + \text{Uniform}(-0.3, 0.3)$
$z^p_{u,i,j}$	$0.5 \cdot u$
$l^p_{u,i,j}$	$\text{Uniform}(0.2, 0.4)$
$w^p_{u,i,j}$	$\text{Uniform}(0.2, 0.4)$
$h^p_{u,i,j}$	$(2u-1) \cdot \text{Uniform}(0.15, 0.35)$

TABLE I: Pyramid parameters

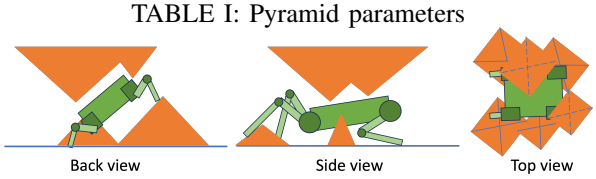

Fig. 3: Different views of a legged robot moving through confined 3D environments composed of random pyramids.

sampled from the range of $[0.2, 0.4]$. The height of the tip is randomly sampled from the range of $[0.15, 0.35]$. Notably, the height of the tip is negative for pyramids on the ceiling to reflect their inverted orientation.

These 3D environments are categorized into three difficulty levels: *easy*, *medium*, and *hard*, with the number of columns n_c set to 2, 3, and 4, respectively. The number of rows, denoted as n_r, remains constant at 3. The size of the pyramid matrices remains the same for all three difficulty levels to cover an area of 2.5 meters by 1.5 meters. Consequently, environments with more columns or rows include denser pyramids, introducing greater challenges for traversal. To navigate these 3D obstacles, the robots are initialized at the point $(-1.75, 0)$, positioning itself 0.5 meters away from the pyramids. The navigation task is to reach the goal location at $(+1.75, 0)$, situated 0.5 meters beyond the pyramids. Fig. 3 shows different views of a robot navigating in a 3D confined environment.

B. Training Details

The implementation of the three approaches is based on the massive parallel training pipeline introduced by Margolis and Agrawal [33], employing the Isaac Gym [34] simulation and the Proximal Policy Optimization (PPO) algorithm [35] as the core Deep RL technique. The policies are trained in parallel to control a Unitree Go1 robot to navigate on 1000 *easy*, 1500 *medium*, and 1500 *hard* environments describe in Section IV-A. Fig 4 visualizes the massive parallel training in Isaac Gym.

a) Domain randomization for Sim-to-real transfer: For better sim-to-real transfer, we randomize a wide range of parameters during training, including the robot's body mass, motor strength, joint position calibration, ground friction and restitution, and orientation and magnitude of gravity.

b) Curriculum: Due to the sparse reward used by the end-to-end dexterity approach, we employ a curriculum strategy. This strategy initiates training with an initial x-coordinate goal location, set at $x_g = 0.6$. Then, the value is incremented by 0.2 meters whenever the success rate of reaching the current goal locations surpasses a threshold of 40%. Importantly, this curriculum strategy is specifically applied to the end-to-end dexterity approach and is not employed for the other two approaches.

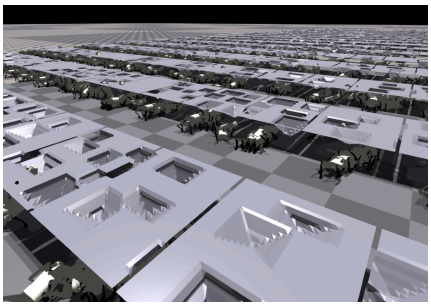

Fig. 4: Massive parallel training in Isaac Gym.

End-to-end dexterity	Hierarchical dexterity	Parameterized motor skills
600 million	**40 million**	80 million

TABLE II: Number of time steps to convergence

V. Results

The trained policies are tested on both simulated and real Go1 robots. This section describes the testing results from these two settings.

A. Simulated results

The trained policies are evaluated on 100 *easy*, 100 *medium*, and 100 *hard* environments. The evaluation runs 10 independent trials in the simulation for each environment, resulting in a total of 3000 test trials. The approximate training steps required for convergence, average success rates during testing, and average collision counts of all successful test trials are reported in Tables II, III, and IV respectively. We provide in-depth analysis as follows.

1) Learning Efficiency: As shown in Table II, during training, the hierarchical dexterity approach demonstrates significantly superior learning efficiency. It converges after approximately 40 million environment steps, whereas the end-to-end dexterity approach requires roughly 600 million environment steps. The simulation runs at a speed of about 5000 steps per second on a single-GPU machine, which attributes to 2.2 hours and 33.3 hours of training for hierarchical dexterity and end-to-end dexterity respectively.

Approach	Easy	Medium	Hard
End-to-end dexterity	29.4%	7.0%	5.1%
Hierarchical dexterity (ours)	**96.8%**	**51.5%**	**39.4%**
Parameterized motor skills	65.7%	4.0%	2.4%

TABLE III: Success rates of the three approaches on confined 3D environments with three difficulty levels.

Approach	Easy	Medium	Hard
End-to-end dexterity	7.9	**16.2**	**29.8**
Hierarchical dexterity	**3.8**	32.3	43.6

TABLE IV: Average Collision Count. Bold numbers indicate better collision avoidance.

We attribute the enhanced performance of the hierarchical dexterity approach to its distinct separation of long-term navigation and short-term motor skill acquisition, in contrast to the end-to-end version. To be specific, the end-to-end training approach attempts to encompass the entire solution space for dexterous locomotion in confined 3D spaces. However, such an approach is shown to be highly inefficient in our experiments, particularly when dealing with sparse rewards. In contrast, the hierarchical dexterity approach preserves the solution space to effectively explore motor skills but at the same time only needs to reach short-term local goals computed by a classical planner. Such a hierarchical approach provides denser reward signals and therefore facilitates more efficient learning of the motor skills which are essential for successful traversal within confined 3D spaces.

Acquiring predefined motor skills takes approximately 80 million environment steps, which is inferior but comparable in terms of sample efficiency to the hierarchical dexterity approach. However, these predefined motor skills do not enable sufficient dexterity in confined 3D spaces.

2) Learned Dexterity: As shown in Table III, the hierarchical dexterity approach outperforms the other two approaches for all three difficulty levels and consistently maintain a significant advantage. In *easy* environments, the hierarchical dexterity approach almost finishes every single trial, while the parameterized motor skills approach achieves more than twice the success rate of end-to-end learning. It is likely that the limited solution space provided by the parameterized motor skills is sufficient to go through sparse 3D obstacles. Challenges arise when transitioning to more demanding environments, characterized as *medium* or *hard*, which necessitate advanced motor skills beyond those predefined ones. In such scenarios, the success rate experiences a significant decline, underscoring the need for end-to-end or hierarchical training to preserve the full solution space for an agent to explore its own motor skills. The success rates of the end-to-end approach and parameterized motor skills approach both drop to single digits for *medium* and lower single digits for *hard*. Although performance also drops for the hierarchical dexterity approach, it can still finish half of the environments in *medium* and more than one third in *hard*.

3) Obstacle-Aided Locomotion: We also show the average collision count measured during all successful trials of the end-to-end and hierarchical dexterity approaches in Table IV, which reveals an interesting finding in terms of obstacle-aided locomotion. While the collision count for end-to-end dexterity increases with more difficult environments, it does not change much for hierarchical dexterity. Overall, hierarchical dexterity has higher collision count compared to end-to-end learning. Such an observation shows that hierarchical dexterity is more willing to touch obstacles and still achieves a higher success rate than its more conservative counterpart. Considering the all-around obstacle constraints in confined 3D spaces, it is likely that the robot has to lean against some obstacles to assure torso stability and forward progress. Guided by a local goal close to the robot, it can focus more on discovering these emergent motor skills, e.g., obstacle-aided locomotion despite the small collision penalty, to utilize the environment structure to move, rather than trying to figure out how to navigate to a global goal far away from the robot.

B. Real-World Demonstration

We demonstrate the policy learned in simulation on a physical Unitree Go1 quadruped robot by creating an artificial setup replicating the simulated environments (Fig. 1). This setup allows us to assess the policy's performance in a controlled real-world scenario. The robot generates a point cloud from its front-facing stereo camera, which is used to build real-time height maps during deployment. We filter out the points to limit the robot's observation to cover only a $1\text{m}\times1\text{m}\times0.5\text{m}$ space directly in front of it. Height maps of the ceiling and floor are then generated from the filtered points based on the pitch and roll angle of the camera detected by an onboard IMU and passed to the policy. To enhance the quality of the height maps and mitigate potential noise resulting from the stereo camera's point cloud generation process, we apply some basic smoothing. In this experiment, we configure the goal location to always be one meter in front of the robot to ensure continuous forward motion. Our real-world demonstration shows that the robot can navigate through confined 3D spaces, while occasionally touching the obstacles made from rocks and cardboard boxes, using the controller learned in simulation.

VI. Conclusions and Future Work

This paper introduces a novel and challenging scenario in the realm of legged locomotion. In this scenario, legged robots are tasked with traversing confined 3D spaces that impose constraints from all around the robot, demanding the adaptation of acyclic and asymmetric locomotion behaviors. To tackle this challenging scenario, three distinct approaches are proposed in this paper: end-to-end dexterity, hybrid dexterity, and predefined motor skills. Empirical results point to the hybrid dexterity approach as the most effective and promising solution for locomotion in confined 3D spaces. This approach combines high-level pose planning from a classical planner with low-level local goal-reaching via an RL-based locomotion controller.

While the experiments conducted in this paper utilize randomly generated pyramid environments, it's important to acknowledge that these environments are artificial and may not accurately represent real-world 3D confined spaces, such as tunnels or other complex structures. In the future, exploring training on real-world tunnels or constructing more realistic simulation environments to capture a wider variety of confined 3D spaces remains an important direction.

Furthermore, the paper's focus on a simple reactive planner is noteworthy. This planner may not be able to adequately address scenarios with highly complex navigation paths. An open question remains whether it is feasible to employ more advanced path planning techniques specifically designed for 3D spaces, while ensuring compatibility with the massive training pipeline for legged locomotion in IsaacGym.

REFERENCES

[1] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[2] D. Owaki and A. Ishiguro, "A quadruped robot exhibiting spontaneous gait transitions from walking to trotting to galloping," *Scientific reports*, vol. 7, no. 1, p. 277, 2017.

[3] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 932–950, 2013.

[4] M. A. Lewis and G. A. Bekey, "Gait adaptation in a quadruped robot," *Autonomous robots*, vol. 12, pp. 301–312, 2002.

[5] Y. Shao, Y. Jin, X. Liu, W. He, H. Wang, and W. Yang, "Learning free gait transition for quadruped robots via phase-guided controller," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1230–1237, 2021.

[6] Z. Fu, A. Kumar, J. Malik, and D. Pathak, "Minimizing energy consumption leads to the emergence of gaits in legged robots," *arXiv preprint arXiv:2111.01674*, 2021.

[7] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[8] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," in *Conference on Robot Learning*. PMLR, 2020, pp. 1–10.

[9] X. Pan, T. Zhang, B. Ichter, A. Faust, J. Tan, and S. Ha, "Zero-shot imitation learning from demonstrations for legged robot visual navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 679–685.

[10] 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.

[11] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *arXiv preprint arXiv:2205.02824*, 2022.

[12] T.-Y. Yang, T. Zhang, L. Luu, S. Ha, J. Tan, and W. Yu, "Safe reinforcement learning for legged locomotion," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2454–2461.

[13] 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.

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

[15] 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.

[16] 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 *5th Annual Conference on Robot Learning*, 2021.

[17] T. Rouček, M. Pecka, P. Čížek, T. Petříček, J. Bayer, V. Šalanskỳ, D. Heřt, M. Petrlík, T. Báča, V. Spurnỳ *et al.*, "Darpa subterranean challenge: Multi-robotic exploration of underground environments," in *Modelling and Simulation for Autonomous Systems: 6th International Conference, MESAS 2019, Palermo, Italy, October 29–31, 2019, Revised Selected Papers 6*. Springer, 2020, pp. 274–290.

[18] C. D. Bellicoso, M. Bjelonic, L. Wellhausen, K. Holtmann, F. Günther, M. Tranzatto, P. Fankhauser, and M. Hutter, "Advances in real-world applications for legged robots," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1311–1326, 2018.

[19] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017.

[20] A. A. Transeth, R. I. Leine, C. Glocker, K. Y. Pettersen, and P. Liljebäck, "Snake robot obstacle-aided locomotion: Modeling, simulations, and experiments," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 88–104, 2008.

[21] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 822–10 825, 2008.

[22] 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 *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 38–44.

[23] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6295–6301.

[24] J. Di Carlo, "Software and control design for the mit cheetah quadruped robots," Ph.D. dissertation, Massachusetts Institute of Technology, 2020.

[25] D. Wooden, M. Malchano, K. Blankespoor, A. Howardy, A. A. Rizzi, and M. Raibert, "Autonomous navigation for bigdog," in *2010 IEEE international conference on robotics and automation*. Ieee, 2010, pp. 4736–4741.

[26] S. H. Jeon, S. Kim, and D. Kim, "Online optimal landing control of the mit mini cheetah," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 178–184.

[27] M. Geisert, T. Yates, A. Orgen, P. Fernbach, and I. Havoutis, "Contact planning for the anymal quadruped robot using an acyclic reachability-based planner," in *Towards Autonomous Robotic Systems: 20th Annual Conference, TAROS 2019, London, UK, July 3–5, 2019, Proceedings, Part I*. Springer, 2019, pp. 275–287.

[28] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

[29] R. Buchanan, L. Wellhausen, M. Bjelonic, T. Bandyopadhyay, N. Kottege, and M. Hutter, "Perceptive whole-body planning for multilegged robots in confined spaces," *Journal of Field Robotics*, vol. 38, no. 1, pp. 68–84, 2021.

[30] G. Bellegarda, Y. Chen, Z. Liu, and Q. Nguyen, "Robust high-speed running for quadruped robots via deep reinforcement learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 364–10 370.

[31] Y. Jin, X. Liu, Y. Shao, H. Wang, and W. Yang, "High-speed quadrupedal locomotion by imitation-relaxation reinforcement learning," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1198–1208, 2022.

[32] M. Saggar, T. D'Silva, N. Kohl, and P. Stone, "Autonomous learning of stable quadruped locomotion," in *RoboCup 2006: Robot Soccer World Cup X 10*. Springer, 2007, pp. 98–109.

[33] 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.

[34] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu, A. Handa, and D. Fox, "Factory: Fast contact for robotic assembly," in *Robotics: Science and Systems*, 2022.

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