Supplementary Material
Enabling Behavioral Diversity for Learning Locomotion in Humanoid Robot

Author Names Omitted for Anonymous Review.

This document provides supplementary materials and extended analysis to support the claims and details of the approach *Discovery and Learning of Diverse Locomotion* (DLDL), presented in the main paper *Enabling Behavioral Diversity for Learning Locomotion in Humanoid Robots*. This supplement is organized as follows: Section I discusses the limitations of our work and potential directions for future research. Section II presents the full mathematical derivations for the losses used in the policy training. Furthermore, we include the DLDL training algorithm and the mathematical derivation of its computational complexity. Section III outlines the implementation specifics, including the simulation environment for evaluation, hyperparameter values used in the training algorithm, the details of evaluation metrics calculation, and the implementation of the policy deployment. Finally, Section IV includes additional qualitative results of DLDL, and the discussion of different baseline controller implementation designs.

TABLE I
MATHEMATICAL NOTATIONS.

Variable	**Definition**
$\mathbf{s}$	State
$\mathbf{a}$	Action
$\mathbf{z}$	Latent Variable
$\mathbf{D}$	Offline dataset
$\mathcal{L}$	Loss
Θ	Parameters of the inference network
r	Reward
ϵ	Constraint for KL divergence
λ	Coefficient for balancing mutual information loss
α	Coefficient for balancing data learning
μ	Lagrangian multiplier
Function	**Definition**
$A(\mathbf{s}, \mathbf{a}, \mathbf{z})$	Advantage function
$V(\mathbf{s}, \mathbf{a}, \mathbf{z})$	State-value function
$Q(\mathbf{s}, \mathbf{a}, \mathbf{z})$	Action-value function
$W(\mathbf{s}, \mathbf{a}, \mathbf{z})$	Weighted function
$\beta(\mathbf{a}\|\mathbf{s}, \mathbf{z})$	Underlying behavior policy
$\pi(\mathbf{a}\|\mathbf{s}, \mathbf{z})$	Learned policy
$q_\Theta(\mathbf{z}\|\mathbf{s}, \mathbf{a})$	Posterior distribution of latent variables
$l_\Theta(\mathbf{s}, \mathbf{a}\|\mathbf{z})$	Likelihood of state-action pairs

I. LIMITATIONS AND FUTURE WORKS

While our DLDL demonstrates the capability of learning a humanoid locomotion policy with generating diverse behaviors, we acknowledge several limitations that suggest directions for future research.

First, our DLDL is relying on the datasets containing diverse locomotion behaviors. In particular, learning diverse behaviors on high-degree-of-freedom (DoF) humanoid robots requires datasets that capture a wide range of locomotion behaviors. However, collecting such diverse data is especially challenging due to the complexity of humanoid platforms, the high cost of physical trials, and the difficulty in manually designing or scripting varied behaviors at scale. One direction to tackle this challenge is to introduce methods to generate synthetic or simulated datasets that approximate real-world diversity, such as domain randomization, procedural generation of environments and tasks, or leveraging large-scale motion capture datasets.

The second limitation, common to the humanoid robotics community, is the simulation-to-reality gap. Although modern physics-based simulators like MuJoCo, Genesis, and Isaac Sim have greatly bridged the gap between policy training in simulation and real-world application, significant challenges remain. Discrepancies arising from the complex dynamics of humanoid platforms are difficult to model perfectly, particularly inaccurate contact and friction models, unmodeled actuator dynamics such as motor delays and backlash, and the absence of realistic sensor noise and latency. These problems can cause a policy to learn behaviors that exploit simulation-specific artifacts, negatively impacting its performance when deployed on a physical robot. To address this challenge, methods like fine-tuning policies learned from offline datasets through online training could allow the robot to adapt to the specific dynamics of the physical hardware.

The third limitation is that our approach treats the robot's action space as a flat vector, where the physical relationships between different joints are not explicitly modeled but must be learned implicitly from the dataset. This will limit the policy's ability to generate highly coordinated, whole-body controls, as it has no built-in knowledge of the robot's kinematic chain. A future direction is to incorporate a physics-informed module into the learning architecture, such as graph neural network (GNN). By representing the humanoid as a graph of links and joints, a GNN-based policy could more efficiently learn coordinated movements for humanoids as it can explicitly mirrors the physical structure of the robot.

II. MATHEMATICAL DERIVATIONS OF BOUNDS AND LOSSES

This section first lists the variables and mathematical notations in Table II used in our mathematical expressions and

derivations, and then provides the mathematical derivations of the variational lower bounds and loss functions.

A. Derivation of the Variational Lower Bound of Mutual Information

We show the derivation from Eq. (2) to Eq. (4). We first derive the expected KL divergence between $p(\mathbf{z}|\mathbf{s},\mathbf{a})$ and $p(\mathbf{z})$ by the following:

$$\max\ I(\mathbf{z};\mathbf{s},\mathbf{a}) \tag{1}$$
$$=\max\ D_{\mathrm{KL}}(p(\mathbf{z},\mathbf{s},\mathbf{a})\|p(\mathbf{z})p(\mathbf{s},\mathbf{a})) \tag{2}$$
$$=\max\ \mathbb{E}_{p(\mathbf{s},\mathbf{a},\mathbf{z})}\left[\log\frac{p(\mathbf{z}|\mathbf{s},\mathbf{a})p(\mathbf{s},\mathbf{a})}{p(\mathbf{s},\mathbf{a})p(\mathbf{z})}\right] \tag{3}$$
$$=\max\ \mathbb{E}_{p(\mathbf{s},\mathbf{a},\mathbf{z})}\left[\log\frac{p(\mathbf{z}|\mathbf{s},\mathbf{a})}{p(\mathbf{z})}\right] \tag{4}$$
$$=\max\ \mathbb{E}_{p(\mathbf{s},\mathbf{a})}\left[\mathbb{E}_{p(\mathbf{z}|\mathbf{s},\mathbf{a})}\left[\log\frac{p(\mathbf{z}|\mathbf{s},\mathbf{a})}{p(\mathbf{z})}\right]\right] \tag{5}$$
$$=\max\ \mathbb{E}_{p(\mathbf{s},\mathbf{a})}\left[D_{\mathrm{KL}}\left(p(\mathbf{z}|\mathbf{s},\mathbf{a})\|p(\mathbf{z})\right)\right] \tag{6}$$

In this derivation, we apply the chain rule of probability to Eq. (3) and Eq. (4) and derive the expectation in Eq. (5). We finally represent Eq. (5) as the KL divergence in Eq. (6).

Since the latent variable distribution $p(\mathbf{z})$ is intractable, we introduce the posterior $q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})$ to compute the variational lower bound of the expected KL divergence Eq. (6). Our derivation is shown as the following:

$$\mathbb{E}_{p(\mathbf{s},\mathbf{a})}\left[D_{\mathrm{KL}}\left(p(\mathbf{z}|\mathbf{s},\mathbf{a})\|p(\mathbf{z})\right)\right] \tag{7}$$
$$\begin{aligned}=&\mathbb{E}_{p(\mathbf{s},\mathbf{a},\mathbf{z})}[\log p(\mathbf{z}|\mathbf{s},\mathbf{a})-\log q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})\\&+\log q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})-\log p(\mathbf{z})]\end{aligned} \tag{8}$$
$$\begin{aligned}=&\mathbb{E}_{p(\mathbf{s},\mathbf{a})}\left[D_{\mathrm{KL}}(p(\mathbf{z}|\mathbf{s},\mathbf{a})\|q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a}))\right]\\&+\mathbb{E}_{p(\mathbf{s},\mathbf{a},\mathbf{z})}\left[D_{\mathrm{KL}}(q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})\|p(\mathbf{z}))\right]\end{aligned} \tag{9}$$
$$\geq\mathbb{E}_{p(\mathbf{s},\mathbf{a},\mathbf{z})}\left[\log q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})\right]-\mathbb{E}_{p(\mathbf{z})}\left[\log p(\mathbf{z})\right] \tag{10}$$
$$=\mathbb{E}_{p(\mathbf{s},\mathbf{a},\mathbf{z})}\left[\log q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})\right]+H(\mathbf{z}) \tag{11}$$
$$\geq\mathbb{E}_{p(\mathbf{s},\mathbf{a},\mathbf{z})}\left[\log q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})\right] \tag{12}$$

After the introduction of $q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})$ in Eq. (8), the expression is then regrouped in Eq. (9) to formulate two KL divergence terms. By dropping the first term, a lower bound is established in Eq. (10). Then, We simplify this expression by identifying the entropy of the prior $H(\mathbf{z})$ in Eq. (11). Finally, since $H(\mathbf{z})$ is constant with respect to the learning process, it is removed to yield the final, tractable lower bound in Eq. (12).

B. Derivation of the Loss for Stable Posterior Learning

We derive the loss Eq. (11) for learning posterior by following the method introduced by AWAC [1] and DiveOff [2]. We start from introducing a normalization constraint to the objective for stable posterior learning, which ensures that the encoder network can model a valid posterior distribution for different latent variables instead of modeling a biased posterior for only a single latent variable. We formulate the problem as:

$$\max_{q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})}\mathbb{E}_{\mathbf{z}\sim q_\Theta(\cdot|\mathbf{s},\mathbf{a})}[A(\mathbf{s},\mathbf{a},\mathbf{z})] \tag{13}$$
$$\text{s.t. } D_{\mathrm{KL}}\left(q_\Theta(\cdot|\mathbf{s},\mathbf{a})\ ||\ q_{\text{old}}(\cdot|\mathbf{s},\mathbf{a})\right)\leq\epsilon_q \tag{14}$$
$$\int q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})=1 \tag{15}$$

Then, we formulate the Lagrangian of Eq. (13) as:

$$\begin{aligned}L(q_\Theta,\alpha_q,\mu_q)=&\mathbb{E}_{\mathbf{z}\sim q_\Theta(\cdot|\mathbf{s},\mathbf{a})}[A(\mathbf{s},\mathbf{a},\mathbf{z})]\\&-\alpha_q\left(D_{\mathrm{KL}}(q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})\|q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a}))\right)\\&+\mu_q\left(\int q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})d\mathbf{z}-1\right)\end{aligned} \tag{16}$$

Then, we calculate the derivative of Eq. (16) with respect to $q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})$ and present it as:

$$\frac{\partial L(q_\Theta,\alpha_q,\mu_q)}{\partial q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})}=A(\mathbf{s},\mathbf{a},\mathbf{z})-\alpha_q\left(\log\frac{q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})}{q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})}+1\right)+\mu_q \tag{17}$$

We set the derivative Eq. (17) to zero and solve it to find the extrema of this function. We present the solution as:

$$q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})=q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})\exp\left(\frac{A(\mathbf{s},\mathbf{a},\mathbf{z})}{\alpha_q}\right)\exp\left(\frac{\mu_q}{\alpha_q}-1\right) \tag{18}$$
$$\propto q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})\exp\left(\frac{1}{\alpha_q}A(\mathbf{s},\mathbf{a},\mathbf{z})\right) \tag{19}$$

We ignore the term $\exp(\frac{\mu_q}{\alpha_q})-1$ in Eq. (18) since it is constant with respect to $\mathbf{z}$, yielding the final solution in Eq. (19). We minimize the KL divergence between the posterior $q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})$ and the posterior proportional to the extrema point $q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})\exp\left(\frac{1}{\alpha_q}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)$, yielding the following equations:

$$D_{\mathrm{KL}}(q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})\exp\left(\frac{1}{\alpha_q}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)\|q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})) \tag{20}$$
$$\begin{aligned}=&\int q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})\exp\left(\frac{1}{\alpha_q}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)\\&\log\left(\frac{q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})\exp\left(\frac{1}{\alpha_q}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)}{q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})}\right)d\mathbf{z}\end{aligned} \tag{21}$$
$$\begin{aligned}=&\int q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})\exp\left(\frac{1}{\alpha_q}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)\log q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})d\mathbf{z}\\&+\frac{1}{\alpha_q}\int q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})\exp\left(\frac{1}{\alpha_q}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)A(\mathbf{s},\mathbf{a},\mathbf{z})d\mathbf{z}\\&-\int q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})\exp\left(\frac{1}{\alpha_q}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)\log q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})d\mathbf{z}\end{aligned} \tag{22}$$

Since the first and second term in Eq. (22) is independent with the posterior $q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})$, we can ignore them and formulate the loss for learning posterior with the third term in Eq. (22):

$$\max_{q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})}\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathbf{D}}\mathbb{E}_{\mathbf{z}\sim q_{\text{old}}(\cdot|\mathbf{s},\mathbf{a})}[W_q(\mathbf{s},\mathbf{a},\mathbf{z})\log q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})] \tag{23}$$

We let $W_q(\mathbf{s},\mathbf{a},\mathbf{z}) = q_{\text{old}}(\mathbf{z}|\mathbf{s},\mathbf{a})\exp\left(\frac{1}{\alpha_q}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)$, yielding the loss for stable posterior learning.

C. Derivation of the Loss for Behavioral-Diverse Policy Learning

The loss Eq. (16) for learning the policy $\pi(\mathbf{a}|\mathbf{s},\mathbf{z})$ is derived using the same procedure in II-B. We begin with introducing a normalization constraint, ensuring the learned policy network can model the distributions of different actions, to the objective for policy learning and formulate the following problem:

$$\max_{\pi(\mathbf{a}|\mathbf{s},\mathbf{z})} \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s},\mathbf{z})}[A(\mathbf{s},\mathbf{a},\mathbf{z})] \tag{24}$$

$$\text{s.t. } D_{\text{KL}}\left(\pi(\cdot|\mathbf{s},\mathbf{z}) \mid\mid \beta(\cdot|\mathbf{s},\mathbf{z})\right) \leq \epsilon_\pi \tag{25}$$

$$\int \pi(\mathbf{a}|\mathbf{s},\mathbf{z})d\mathbf{a} = 1 \tag{26}$$

Then, we formulate the Lagrangian of Eq. (24) as:

$$\begin{aligned} L(\pi,\alpha_\pi,\mu_\pi) =& \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s},\mathbf{z})}[A(\mathbf{s},\mathbf{a},\mathbf{z})] \\ &- \alpha_\pi\left(D_{\text{KL}}(\pi(\mathbf{a}|\mathbf{s},\mathbf{z})|\beta(\mathbf{a}|\mathbf{s},\mathbf{z})) - \epsilon_\pi\right) \\ &+ \mu_\pi\left(\int \pi(\mathbf{a}|\mathbf{s},\mathbf{z})d\mathbf{a} - 1\right) \end{aligned} \tag{27}$$

Next, we calculate the functional derivative of Eq. (27) with respect to $\pi(\mathbf{a}|\mathbf{s},\mathbf{z})$:

$$\frac{\partial L(\pi,\alpha_\pi,\mu_\pi)}{\partial\pi(\mathbf{a}|\mathbf{s},\mathbf{z})} = A(\mathbf{s},\mathbf{a},\mathbf{z}) - \alpha_\pi\left(\log\frac{\pi(\mathbf{a}|\mathbf{s},\mathbf{z})}{\beta(\mathbf{a}|\mathbf{s},\mathbf{z})} + 1\right) + \mu_\pi \tag{28}$$

We set the derivative to zero and find the extrema with respect to $\pi(\mathbf{a}|\mathbf{s},\mathbf{z})$, yielding the solution:

$$\pi(\mathbf{a}|\mathbf{s},\mathbf{z}) = \beta(\mathbf{a}|\mathbf{s},\mathbf{z})\exp\left(\frac{A(\mathbf{s},\mathbf{a},\mathbf{z})}{\alpha_\pi}\right)\exp\left(\frac{\mu_\pi}{\alpha_\pi} - 1\right) \tag{29}$$

$$\propto \beta(\mathbf{a}|\mathbf{s},\mathbf{z})\exp\left(\frac{1}{\alpha_\pi}A(\mathbf{s},\mathbf{a},\mathbf{z})\right) \tag{30}$$

We ignore the term $\exp\left(\frac{\mu_\pi}{\alpha_\pi} - 1\right)$ in Eq. (29) as it is constant. Then, we formulate an objective to minimize the KL divergence between our learnable policy $\pi(\mathbf{a}|\mathbf{s},\mathbf{z})$ and the optimal solution $\beta(\mathbf{a}|\mathbf{s},\mathbf{z})\exp\left(\frac{1}{\alpha_\pi}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)$, yielding the following equations:

$$D_{\text{KL}}(\beta(\mathbf{a}|\mathbf{s},\mathbf{z})\exp\left(\frac{1}{\alpha_\pi}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)\|\pi(\mathbf{a}|\mathbf{s},\mathbf{z})) \tag{31}$$

$$\begin{aligned} =& \int \beta(\mathbf{a}|\mathbf{s},\mathbf{z})\exp\left(\frac{1}{\alpha_\pi}A(\mathbf{s},\mathbf{a},\mathbf{z})\right) \\ & \log\left(\frac{\beta(\mathbf{a}|\mathbf{s},\mathbf{z})\exp\left(\frac{1}{\alpha_\pi}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)}{\pi(\mathbf{a}|\mathbf{s},\mathbf{z})}\right)d\mathbf{a} \end{aligned} \tag{32}$$

$$\begin{aligned} =& \int \beta(\mathbf{a}|\mathbf{s},\mathbf{z})\exp\left(\frac{1}{\alpha_\pi}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)\log\beta(\mathbf{a}|\mathbf{s},\mathbf{z})d\mathbf{a} \\ &+ \frac{1}{\alpha_\pi}\int \beta(\mathbf{a}|\mathbf{s},\mathbf{z})\exp\left(\frac{1}{\alpha_\pi}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)A(\mathbf{s},\mathbf{a},\mathbf{z})d\mathbf{a} \\ &- \int \beta(\mathbf{a}|\mathbf{s},\mathbf{z})\exp\left(\frac{1}{\alpha_\pi}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)\log\pi(\mathbf{a}|\mathbf{s},\mathbf{z})d\mathbf{a} \end{aligned} \tag{33}$$

The first and second term in Eq. (33) are independent with the policy $\pi(\mathbf{a}|\mathbf{s},\mathbf{z})$, so we ignore them and take the third term to construct the loss. Let $W_\pi(\mathbf{s},\mathbf{a},\mathbf{z}) = \exp\left(\frac{1}{\alpha_\pi}A(\mathbf{s},\mathbf{a},\mathbf{z})\right)$, we formulate the loss as maximizing the following objective:

$$\max_{\pi(\mathbf{a}|\mathbf{s},\mathbf{z})} \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathbf{D}}\mathbb{E}_{\mathbf{z}\sim q_\Theta(\cdot|\mathbf{s},\mathbf{a})}[W_\pi(\mathbf{s},\mathbf{a},\mathbf{z})\log\pi_\phi(\mathbf{a}|\mathbf{s},\mathbf{z})] \tag{34}$$

Algorithm 1: Joint Training via AWAC

1 Initialize $\pi(\mathbf{a}|\mathbf{s},\mathbf{z})$, $q_\Theta(\mathbf{z}|\mathbf{s},\mathbf{a})$ and $l_\Theta(\mathbf{s},\mathbf{a}|\mathbf{z})$
2 Initialize $Q_1(\mathbf{s},\mathbf{a},\mathbf{z})$ and $Q_2(\mathbf{s},\mathbf{a},\mathbf{z})$
3 **while *not terminated*** **do**
4 Sample a mini-batch $\{(\mathbf{s}_i,\mathbf{a}_i,r_i)\}_{i=1}^n$ from $\mathbf{D}$
5 Sample a latent variable $\mathbf{z}_i \sim q_\Theta(\mathbf{z}_i|\mathbf{s}_i,\mathbf{a}_i)$
6 Sample an policy action $\mathbf{a}' \sim \pi(\mathbf{a}'|\mathbf{s}_i,\mathbf{z}_i)$
7 For all i:
$y_i = r_i + \gamma\min\left(Q_1(\mathbf{s}_i,\mathbf{a}',\mathbf{z}_i), Q_2(\mathbf{s}_i,\mathbf{a}',\mathbf{z}_i)\right)$
8 Compute critic target value:
$t = \frac{1}{n}\sum_{i=1}^n(y_i - Q_j(\mathbf{s}_i,\mathbf{a}_i,\mathbf{z}_i)), j = 1,2$
9 Update critics Q_1, Q_2 with t
10 Compute advantage value $A(\mathbf{s}_i,\mathbf{a}_i,\mathbf{z}_i)$:
$q = \min(Q_1(\mathbf{s}_i,\mathbf{a}_i,\mathbf{z}_i), Q_2(\mathbf{s}_i,\mathbf{a}_i,\mathbf{z}_i))$
$v = \min(Q_1(\mathbf{s}_i,\mathbf{a}',\mathbf{z}_i), Q_2(\mathbf{s}_i,\mathbf{a}',\mathbf{z}_i))$
$A(\mathbf{s}_i,\mathbf{a}_i,\mathbf{z}_i) = q - v$
11 Compute $W_\pi(\mathbf{s}_i,\mathbf{a}_i,\mathbf{z}_i)$ and $W_q(\mathbf{s}_i,\mathbf{a}_i,\mathbf{z}_i)$
12 Update π with $\mathcal{L}_\pi$
13 Update q_Θ, l_Θ and π with $\mathcal{L}_q + \mathcal{L}_l + \lambda\mathcal{L}_I$
14 **return** π

D. Training Algorithm

The algorithm designed to train DLDL is presented in Algorithm.1. After the initialization on Line 1 and Line 2, given a dataset $\mathbf{D}$ as the input, we sample a mini-batch of state, action, and reward data $\{(\mathbf{s},\mathbf{a},r)\}_{i=1}^n$ on Line 4. Then, the latent variable $\mathbf{z}_i$ is sampled from the encoder on Line 5, and an action $\mathbf{a}'$ is sampled given $\mathbf{s}_i$ and $\mathbf{z}_i$ on Line 6. The critic target value t is computed on Lines 7 and 8 and is

used to update the critic network parameters on Line 9. The advantage value A is computed on Line 10 with the action-value function and state-value function estimated by two critics $Q_1(\mathbf{s}, \mathbf{a}, \mathbf{z})$ and $Q_2(\mathbf{s}, \mathbf{a}, \mathbf{z})$. We compute $W_\pi(\mathbf{s}, \mathbf{a}, \mathbf{z})$ and $W_q(\mathbf{s}, \mathbf{a}, \mathbf{z})$ on Line 11 using A. Finally, we compute $\mathcal{L}_\pi$ and $\mathcal{L}_q + \mathcal{L}_l + \lambda\mathcal{L}_I$, and alternatively update π and Θ parameters on Lines 12 and 13, respectively.

E. Complexity Analysis

We analyze the time complexity of Algorithm 1 from the main paper to assess the scalability and efficiency of our DLDL approach, especially with respect to control frequency requirements and its suitability for deployment on resource-constrained humanoid platforms. To compute the time complexity, we calculate the number of basic computational operations required for training, such as multiplication and addition. We first define the variables of the neural networks introduced in our approach, where the inference network, the policy network, and the critic networks are configured with the same structures (e.g. the number of hidden layers, and the dimensionality of input layer, output layer, and hidden layer). We define d^{in} as the dimensionality of the input layer, d^{out} as the dimensionality of the output layer, and h as the dimensionality of a hidden layer, assuming that $d^{\text{in}}, d^{\text{out}} \leq h$. We also define B as the size of a data batch, and c as a constant number representing the cost of computational operations in neural networks.

Before starting the policy training, we need to initialize our inference network, described in Line 1 first. We initialize $q_\Theta(\mathbf{z}|\mathbf{s}, \mathbf{a})$ and $l_\Theta(\mathbf{s}, \mathbf{a}|\mathbf{z})$ by training them with the offline dataset. The cost of initializing $q_\Theta(\mathbf{z}|\mathbf{s}, \mathbf{a})$ is represented as $c \cdot T_{\text{init}} \cdot B \cdot \left[(d_q^{\text{in}} + 1)h + (h+1)d_q^{\text{out}} + h^2 + h\right]$, where T_{init} is the number of training iterations for inference network initialization. Therefore, the computational complexity for initializing the posterior is $\mathcal{O}(T_{\text{init}}Bh^2)$. We use the same method to compute the computational complexity of initializing $l_\Theta(\mathbf{s}, \mathbf{a}|\mathbf{z})$, yielding the result as $\mathcal{O}(T_{\text{init}}Bh^2)$. Therefore, the overall computational complexity of inference network initialization is $\mathcal{O}(T_{\text{init}}Bh^2)$.

After the inference network initialization, we start the main training loop of learning the DLDL policy. This loop consists of the training of the critic networks, the inference network, and the policy network. In each training iteration, the computational cost for training a critic network $Q(\mathbf{s}, \mathbf{a}, \mathbf{z})$ in Line 9 is $c \cdot B \cdot \left[(d_Q^{\text{in}} + 1)h + (h+1)d_Q^{\text{out}} + h^2 + h\right] = \mathcal{O}(Bh^2)$. The inference network is also continuously trained in Line 13, yielding the computational cost $\mathcal{O}(Bh^2)$. The computational cost for training the policy network $\pi(\mathbf{a}|\mathbf{s}, \mathbf{z})$ in Line 12 and Line 13 is formulated as $c \cdot B \cdot \left[(d_\pi^{\text{in}} + 1)h + (h+1)d_\pi^{\text{out}} + h^2 + h\right] = \mathcal{O}(Bh^2)$. Let T_{main} be the number of iterations for the main training loop, then we bound the overall computational complexity of the main algorithm by $c \cdot T_{\text{main}} \cdot Bh^2 = \mathcal{O}(T_{\text{main}}Bh^2)$.

We combine the computation of inference network initialization and DLDL policy training, yielding the overall computational complexity of training DLDL as $\mathcal{O}(T_{\text{init}}Bh^2 + T_{\text{main}}Bh^2)$. In addition, the DLDL policy execution only depends on a single forward inference from the policy network, so the computational cost is presented as $c \cdot \left[(d_\pi^{\text{in}} + 1)h + (h+1)d_\pi^{\text{out}} + h^2 + h\right]$, which is bounded by $\mathcal{O}(h^2)$.

III. Additional Implementation and Evaluation Details

A. Simulation Environment Design

To evaluate humanoid locomotion strategies, we developed a simulated warehouse environment that captures the spatial complexity and constraints commonly encountered in real-world industrial settings. Warehouses present a variety of confined spaces, obstacles, and operational constraints that necessitate diverse locomotion behaviors from humanoid robots.

The environment is structured into three distinct zones, each specifically designed to assess a targeted locomotion capability:

- **Low-Clearance Zone:** This area features a partially open roller shutter that simulates a physical height constraint. To navigate through this space, the humanoid robot must execute a *sneaking* behavior – lowering its body posture to avoid contact with the overhead barrier. This task tests the robots ability to modulate its body height dynamically while maintaining balance and forward motion.
- **Narrow Passage Zone:** Here, a tight corridor is formed between a conveyor belt and stacks of cargo, emulating the kind of cluttered and restrictive spaces often found in operational warehouses. Standard forward walking is impractical in this zone; instead, the humanoid robot is required to perform *side-stepping* motions. This scenario evaluates the robots lateral maneuverability and its ability to maintain stable locomotion in constrained environments.
- **Open Navigation Zone:** This area provides an unobstructed floor space with a cargo box placed arbitrarily to serve as an obstacle. The humanoid robot is tasked with performing *forward walking* while incorporating basic obstacle avoidance strategies. This setup serves as a baseline to validate the robots fundamental walking capabilities and reactive motion planning in the presence of minor environmental interference.

Together, these zones form a comprehensive benchmark that challenges a humanoid robot to switch between distinct locomotion strategies in response to varied spatial constraints, closely mirroring the demands of real-world deployment scenarios in logistics and industrial automation.

We selected the Genesis simulator as our evaluation platform due to its suitability for research in humanoid locomotion. Genesis offers a high-fidelity physics engine capable of accurately modeling the complex, contact-rich dynamics inherent to humanoid motion, such as ground contact, balance control, and joint compliance. This level of physical realism is critical for developing and evaluating robust locomotion behaviors. In addition, Genesis provides

high-throughput simulation performance, which significantly accelerates both training and evaluation phases. This is particularly advantageous when using data-intensive methods such as reinforcement learning or large-scale behavior cloning, where simulation speed can be a major bottleneck. Finally, the simulator offers flexible support for environment customization, allowing us to design and implement the diverse warehouse scenarios required for our experiments. This feature is essential for validating our methods ability to generalize across a range of locomotion tasks and environmental constraints.

B. Hyperparameter Values

TABLE II
HYPERPARAMETERS AND NETWORK STRUCTURES USED IN OUR DLDL IMPLEMENTATION.

Hyperparameter	Value
Learning rate for policy network	3e-4
Learning rate for critic network	3e-4
Learning rate for posterior distribution	3e-4
Learning rate for likelihood	3e-4
Learning rate for mutual information loss $\mathcal{L}_I$	6e-5
Discount factor	0.99
Mini-batch size	256
Update rate for policy network parameters	5e-3
Update rate for critic network parameters	5e-3
Update frequency of policy network parameters	2
Balancing coefficient α_π	1/3
Balancing coefficient α_q	1.0
Balancing coefficient for mutual information λ	2.0
Dimensionality of hidden layers in policy network	1024
Dimensionality of hidden layers in critic network	1024
Dimensionality of encoder network	1024
Dimensionality of decoder network	1024
Dimensionality of latent variable	16
Number of hidden layers in policy network	2
Number of hidden layers in critic network	2
Number of hidden layers in encoder network	2
Number of hidden layers in decoder network	2
Activation function of policy network	ReLU
Activation function of critic network	ReLU
Activation function of encoder network	ReLU
Activation function of decoder network	ReLU

C. Quality-Diversity(QD) Score Calculation

We followed the previous works [3]–[6] to calculate the Quality-Diversity(QD) score. We construct an archive to store the number of survival steps of each diverse locomotion behavior. We define an archive as a grid space with n dimensions. We define $[l_i, u_i], i \in [1, n]$ as the lower and upper bound of the space on i^{th} dimension of the grid space. We uniformly discretize the space on i^{th} dimension into c grids. Given one locomotion evaluation, we take the number of survival steps τ as a score. Then, we compute an index tuple $(b_1, \ldots, b_n), b_i \in [l_i, u_i]$ that characterizes the behavior in this evaluation. This continuous vector is then mapped to a discrete grid in the archive. The score in that grid is updated with τ if the grid is empty or τ is greater than the score currently stored there. Let $\tau_{b_1,\cdots,b_n}$ be the score stored in the grid indexed by $(b_1, \ldots, b_n)$, we then compute the QD score measuring how well a policy can simultaneously generate effective locomotion and diverse behaviors by:

$$QD(\pi) = \sum_{b_1=1}^{c} \cdots \sum_{b_n=1}^{c} \tau_{b_1,\cdots,b_n} \tag{35}$$

where a higher c indicates the higher archive's granularity, allowing more evaluations with subtle varied behaviors are mapped to distinct grids.

In our experiments, we use a 3-dimensional vector where each dimension represents the average linear velocity along a cardinal axis (x, y, z). The index tuple is therefore the average velocity vector (v_x, v_y, v_z), with each component bounded with $[-0.5, 0.5]$.

D. Product of Experts

During execution, the joint target positions generated by the policy are passed to a Product of Experts (PoE) model [7], which combines outputs from a baseline locomotion controller and from a learned policy. The baseline controller outputs the robot locomotion control to ensure the robot will not fall, and the learned policy generates diverse behaviors. The final action is generated by the PoE and is then executed by Genesiss or the robot's built-in PD controller to control the robot.

Given a state $\mathbf{s}$ and a latent variable $\mathbf{z}$, we define a baseline controller as $\Phi(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\mathbf{a}; \mu_\Phi(\mathbf{s}), \Sigma_\Phi)$, where the mean $\mu_\Phi(\mathbf{s})$ is the deterministic action from $\Phi(\mathbf{a}|\mathbf{s})$, and the covariance matrix is defined as $\Sigma_\Phi = \exp(-2.4) \cdot \mathbf{I}$. The policy is defined as $\pi(\mathbf{a}|\mathbf{s}, \mathbf{z}) = \mathcal{N}(\mathbf{a}; \mu_\pi(\mathbf{s}, \mathbf{z}), \Sigma_\pi(\mathbf{s}, \mathbf{z}))$, where the mean $\mu_\pi(\mathbf{s}, \mathbf{z})$ and the covariance $\Sigma_\pi(\mathbf{s}, \mathbf{z})$ are determined by the policy network. We formulate the final policy $\pi_f(\mathbf{a}|\mathbf{s}, \mathbf{z})$ from PoE as:

$$\pi_f(\mathbf{a}|\mathbf{s}, \mathbf{z}) = \mathcal{N}(\mathbf{a}; \mu_f(\mathbf{s}, \mathbf{z}), \Sigma_f(\mathbf{s}, \mathbf{z})) \tag{36}$$

$$\text{where} \quad \Sigma_f(\mathbf{s}, \mathbf{z}) = \left(\Sigma_\Phi^{-1} + \Sigma_\pi^{-1}(\mathbf{s}, \mathbf{z})\right)^{-1} \tag{37}$$

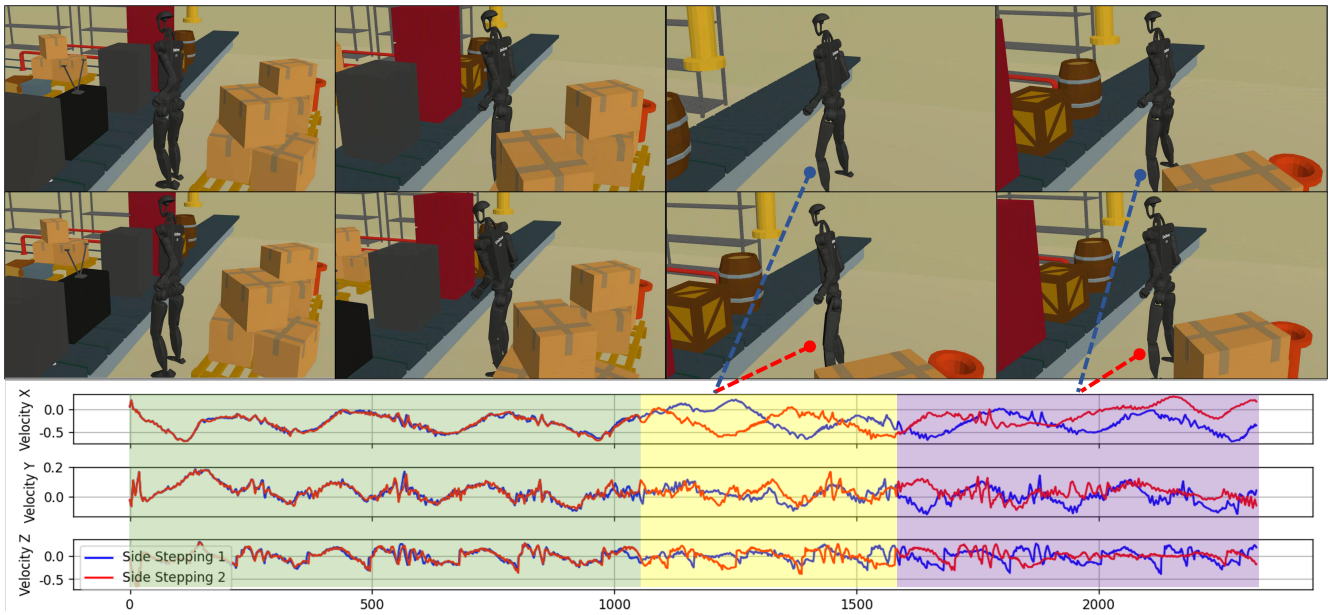
$$\mu_f(\mathbf{s}, \mathbf{z}) = \Sigma_f(\mathbf{s}, \mathbf{z})(\Sigma_\Phi^{-1}\mu_\Phi(\mathbf{s}) + \Sigma_\pi^{-1}(\mathbf{s}, \mathbf{z})\mu_\pi(\mathbf{s}, \mathbf{z})) \tag{38}$$

IV. ADDITIONAL EXPERIMENTAL RESULTS

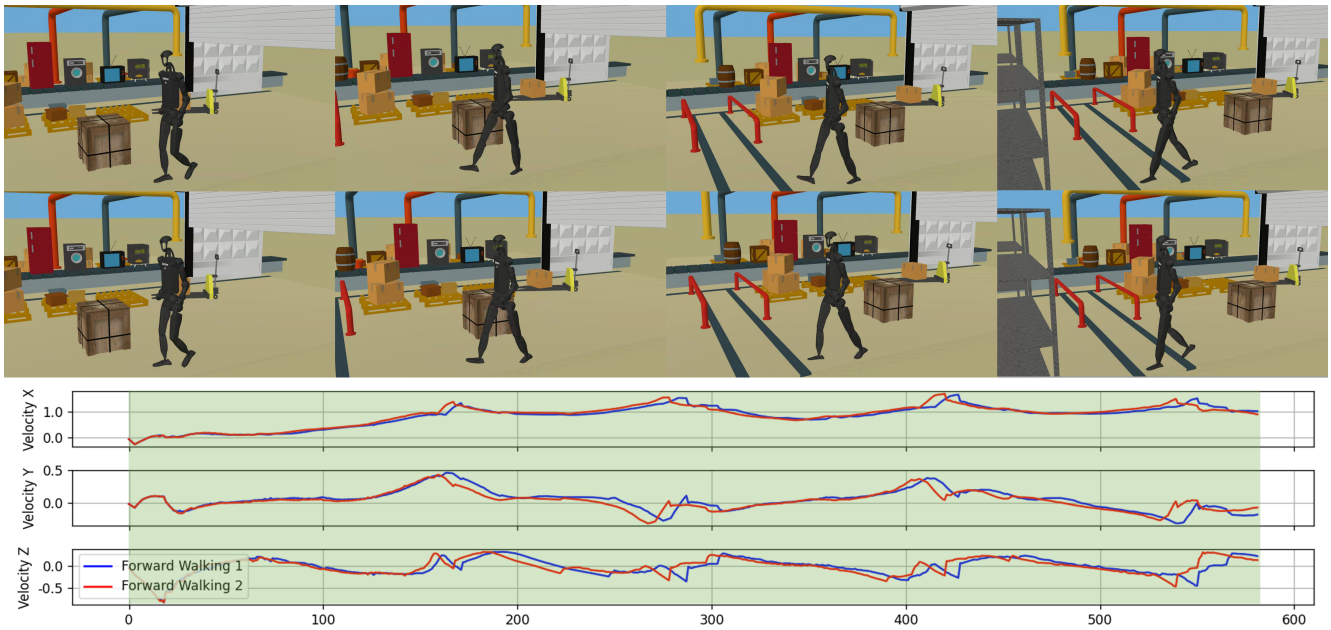
A. Additional Qualitative Results on Diverse Humanoid Locomotion

We perform further experiments using the Unitree H1-2 humanoid robot in Genesis simulations to demonstrate the diverse locomotion behaviors in scenarios of side-stepping and forward walking.

We illustrate a H1-2 humanoid performing side stepping in Fig. 1(a). Two humanoids start from the same initial position and each is initialized with a different latent variable. Their velocity profiles are nearly identical at first, but after step 1100 their behaviors begin to diverge. The first robot pauses in front of the conveyor belt while the second continues moving. After step 1300, the first robot resumes its motion, and the behavioral differences become more pronounced after step 1600. Ultimately, the first robot reaches the end of the conveyor and reverses direction, whereas the second remains stationary for the rest of the evaluation.



(a) The H1-2 humanoid robot learns diverse sneaking behaviors to navigate a narrow passage between a conveyor belt and adjacent cargo piles.



(b) Diverse **forward-walking** behaviors learned by the H1-2 humanoid robot for locomotion in open areas.

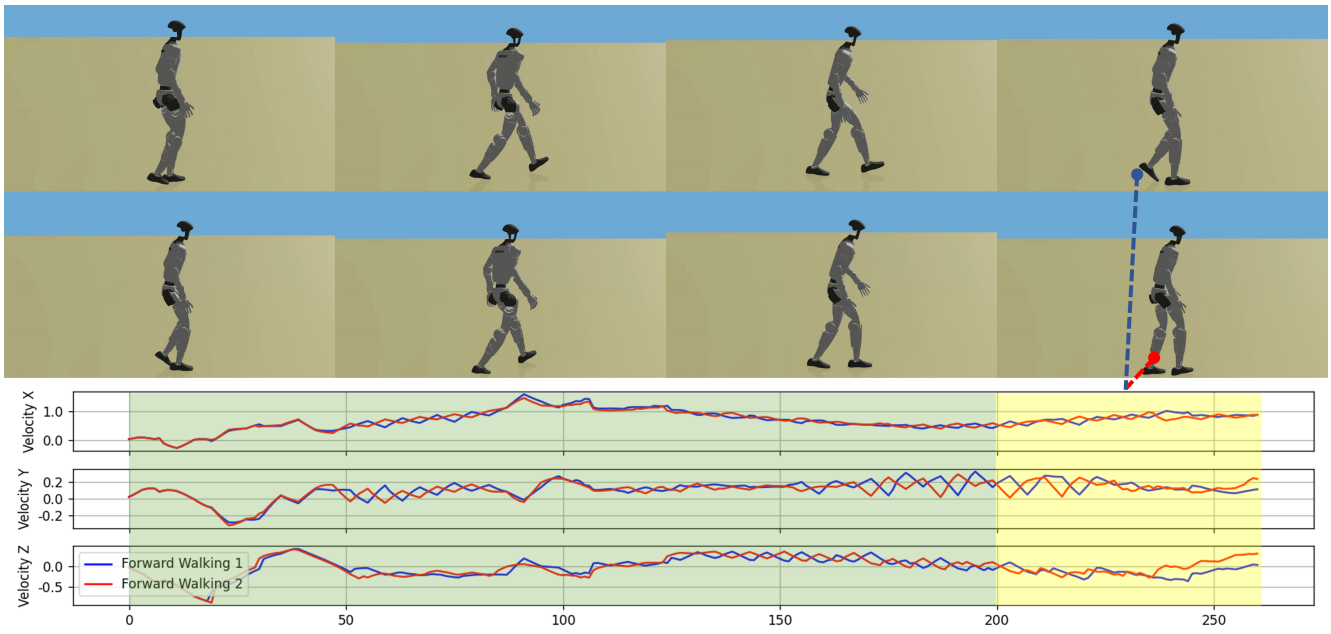
Fig. 1. Additional results obtained by DLDL demonstrate that a H1-2 humanoid robot can learn diverse locomotion behaviors to navigate warehouse environments under varying constraints.

In addition, we present the qualitative results demonstrating H1-2 forward walking locomotion in Fig. 1(b). In this experiment, although the two H1-2 humanoids are conditioned on different latent variables to encourage diversity, they performed the locomotion with nearly identical gaits. We hypothesize that this observed similarity is a result of the short execution time. Since the behavioral divergence driven by the latent variables accumulates gradually, a longer evaluation period is likely required for distinct walking styles to become apparent.

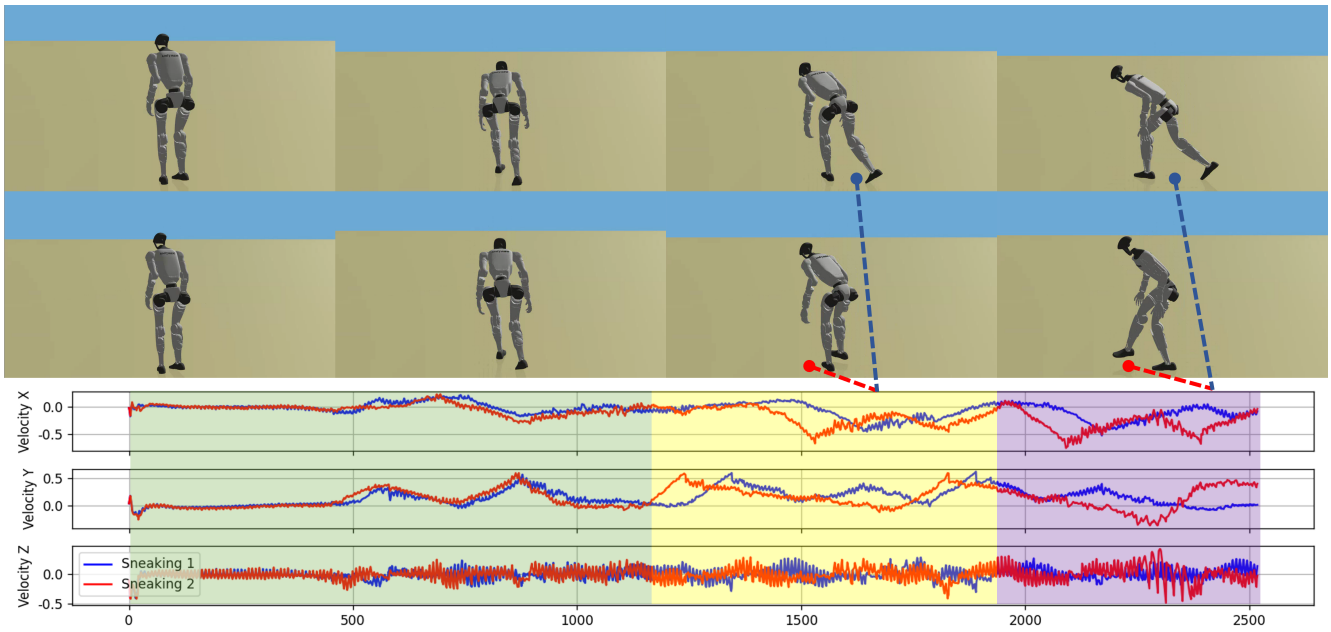
We also include the qualitative results of G1 robot performing sneaking and forward walking in Fig. 2. These results mirror the qualitative results observed on the H1-2 robot. The G1 robot generated clearly diverse behaviors for the sneaking task, while the behavioral divergence during the *forward walking* task was minimal due to the insufficient time for diversity emergence.

B. Discussion of Baseline Controller Covariance

Let the covariance of baseline locomotion controller be $\Sigma_\Phi = \exp(x) \cdot \mathbf{I}$ in our PoE implementation, we provide the effect on QD scores with the difference value settings of x,



(a) Diverse **forward walking** behaviors learned by the G1 Humanoid in open areas.



(b) Diverse **sneaking** behaviors performed by the G1 Humanoid with two different step sizes.

Fig. 2. Qualitative results obtained by DLDL demonstrate that Unitree G1 humanoid can generate diverse locomotion behaviors.

while other hyperparameters remain unchanged. We present the results in Table III.

As we change the value of x and collect 30 evaluations under this setting, the overall QD scores change accordingly. We can observe that, our DLDL approach achieves the highest scores in *forward walking* task with the largest value $x = -1.2$ while it performs the worst in the other two tasks. With $x = -2.4$, DLDL achieves better scores compared to the ones with settings $x = -3.6$ and $x = -4.8$ in *forward walking* task. and outperforms all others in *side stepping* and *sneaking* tasks. When we set $x = -3.6$ and $x = -4.8$, DLDL's scores become lower. We reason that, as the value of x becomes too small, the confidence we assign to the baseline controller is too high. Such assignment can control the robot to perform more effective locomotion without falling, but greatly reduces the behavioral diversity and therefore leads lower QD scores. On the other hand, assigning too large value to x can achieve better behavioral diversity but hinder the effectiveness of locomotion. With $x = -1.2$, DLDL's output can not generate effective locomotion in *side stepping* and *sneaking* so it achieve lower QD scores. We also observe that, in simpler *forward walking* task, our method can generate

TABLE III

OVERALL QD SCORE OF DLDL WITH DIFFERENT BASELINE CONTROLLER COVARIANCE SETTINGS.

Value of x	Forward Walking			Side Stepping			Sneaking		
	$c = 10$	$c = 25$	$c = 50$	$c = 10$	$c = 25$	$c = 50$	$c = 10$	$c = 25$	$c = 50$
$x = -1.2$	**513**	**1595**	**1679**	364	443	845	257	379	994
$x = -2.4$	421	580	901	**4512**	**7501**	**11645**	**3584**	**4125**	**9391**
$x = -3.6$	215	215	430	1878	3011	3011	1274	2548	2548
$x = -4.8$	215	215	430	1133	2266	3011	1274	2548	2548

effective locomotion without sacrificing behavioral diversity. We therefore select $x = -2.4$ as our setting to the baseline controller during evaluation, as it can balance the effective locomotion control and diverse behavior generation.

REFERENCES

[1] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[2] T. Osa and T. Harada, "Discovering multiple solutions from a single task in offline reinforcement learning," in *International Conference on Machine Learning*, 2024.

[3] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[4] M. Flageat, B. Lim, L. Grillotti, M. Allard, S. C. Smith, and A. Cully, "Benchmarking quality-diversity algorithms on neuroevolution for reinforcement learning," *arXiv preprint arXiv:2211.02193*, 2022.

[5] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, p. 40, 2016.

[6] D. Gravina, A. Khalifa, A. Liapis, J. Togelius, and G. N. Yannakakis, "Procedural content generation through quality diversity," in *IEEE Conference on Games*, 2019.

[7] G. E. Hinton, "Products of experts," 1999.