Supplementary Materials for Learning Diverse Natural Behaviors for Enhancing the Agility of Quadrupedal Robots

S1. Nomenclature

$\mathbf{s}$	state
$\mathbf{o}$	observation
$\mathbf{a}$	action
$\mathbf{a}^{\mathrm{D}}$	discrete action
$\mathbf{a}^{\mathrm{C}}$	continuous action
r	reward
π	policy
H	observation horizon
D	discriminator
Q	predictor
$\mathbf{c}$	latent skill variable
ϵ	latent shifting variable
y	behavior label
$\mathcal{H}$	entropy
$I(\cdot)$	mutual information
$(\cdot)^{\mathrm{I}}$	imitation quantity
$(\cdot)^{\mathrm{T}}$	task quantity
$(\cdot)^{\mathrm{S}}$	simulator quantity
$\tilde{(\cdot)}$	real-world quantity
$\hat{(\cdot)}$	predicted quantity
$(\cdot)^{\mathrm{SS}}$	semi-supervised quantity
$(\cdot)^{\mathrm{US}}$	unsupervised quantity
$\mathbb{I}$	indicator function
$\|\|\cdot\|\|$	l_2 norm

S2. Training details

The training of the integrated controller is divided into three stages: BBC pretraining, BBC fine-tuning, and TSC training. All training processes are conducted in the IsaacGym physics simulation environment.

In the first stage, the simulator's physical parameters are set empirically. We use semi-supervised InfoGAIL to train a multimodal BBC based on motion capture data of common dog behaviors, with less than 5% of the data labeled. Aside from motion retargeting, the data requires no additional preprocessing, such as cropping, labeling, or alignment. To address category imbalance, labeled data is simply repeated to rebalance minority categories. The training pseudocode is presented in Algorithm S1.

In each training iteration i, the latent skill variable $\mathbf{c}$ is sampled from a categorical distribution $p_i(\mathbf{c})$, and the latent shifting variable ϵ is drawn from a uniform distribution $p(\epsilon)$. The policy interacts with the environment based on observations and latent variables, generating a trajectory distribution d^{π}. The discriminator D and predictor

Q are updated via gradient descent on Equation (10). The policy π is then optimized using the PPO algorithm [58] to maximize the reward in Equation (11), allowing it to replicate the multimodal behaviors in the motion capture data. The value function V is updated using $TD(\lambda)$.

To align with the imbalanced motion capture data, $p(\mathbf{c})$ is estimated by $Q^{\mathbf{c}}$ and updated using the exponential moving average:

$$p_{i+1}(\mathbf{c}) = \beta p_i(\mathbf{c}) + (1-\beta)\frac{1}{N}\sum_i Q^{\mathbf{c}}(\hat{y}|\mathbf{o}_i^{\mathrm{I}}), \tag{S1}$$

where β is the weight coefficient. The hyperparameters for semi-supervised InfoGAIL are shown in Table S1.

In addition to imitating motion capture data, the BBC is also trained to follow task-related command inputs to enhance its controllability. The command ranges for each behavior mode are shown in Table S2. During training, task commands and latent variables are sampled every six seconds. Additionally, randomly generated rough terrains and random disturbances applied to the robot are introduced to improve the robustness of the BBC.

In the second stage, we use EASI to learn the simulator's parameter distribution from a small set of real-world samples, allowing the simulator to better approximate the real world. The BBC is then fine-tuned in this enhanced simulator. The real-world samples, collected using the pre-trained BBC (or any other controller), consist of 4,000 steps at a control frequency of 50 Hz, providing about 80 seconds of data. The simulator parameter optimization pseudocode is presented in Algorithm S2.

In each generation, N parameter sets are sampled from the distribution $\Xi^{(i)}$, including the PD parameters for the hip, thigh, and calf joints of the quadrupedal robot, while the known component masses remain fixed. The policy π then collects trajectories d^{π} in N parallel simulations with the sampled parameters. A batch of state transitions is subsequently used to update the discriminator $\tilde{D}_k$ to $\tilde{D}_{k+1}$ following Equation (13). Finally, the updated discriminator calculates the rewards $r^{(i)}$, which are used by ES to generate the next parameter distribution $\Xi^{(i+1)}$. The hyperparameters for EASI are shown in Table S3.

In the last stage, the TSC is trained using privileged learning to perform specific tasks. The pseudocode is presented in Algorithm S3. To address the issue of sparse rewards, the teacher policy π^{TSC} is first trained to maximize task reward and imitation reward, with the latter calculated by the discriminator from the BBC. Hybrid-PPO [47] is employed to handle the hybrid action space. Subsequently, the student policy $\hat{\pi}^{\mathrm{TSC}}$ takes depth images as input and outputs actions to mimic the teacher's behavior. The positions of obstacles and the robot are randomly initialized at the beginning of each episode. Additionally, to enhance the robustness of the student policy in real world, we incorporate the BYOL loss [48]. We apply random augmentations to the input depth images, such as rectangular noise, white noise, and Gaussian blur. The hyperparameters for privileged learning are shown in Table S4.

The networks are implemented using Pytorch [62], and are trained on a computer with an i5-12600KF CPU and a Nvidia GTX 4080 GPU. We use the parallelized

physics simulator IsaacGym [51] for training. The BBC is trained using real dog motion capture data from [57], which includes approximately 20 minutes of motion data covering common dog behaviors. Training the BBC takes about 4 days on a single GPU, while the TSC-teacher and TSC-student require approximately 10 hours and 14 hours, respectively.

S3. Deployment details

We deploy our integrated controller on the Unitree Go2 robot, which is equipped with a Jetson Orin NX, an Intel RealSense D435i depth camera [63] and features four limbs comprising 12 joints.

In our real-world experiments, we deploy our integrated controller that incorporates both BBC and TSC on the Jetson Orin NX embedded in the Unitree Go2 robot, achieving stable control frequency at 50 Hz while performing deep image processing.

To ensure that image processing and policy inference are completed within each 20ms cycle, all computations are accelerated using CUDA. Intercommunication among the robot's various sensors and low-level motors is facilitated via the Cyclone DDS and LCM platforms, with the motors executing torque at a rate of 1000 Hz based on Unitree Go2 integrated PD controllers. Depth images are acquired at a sensor resolution of 848×480, and then these images are subsequently downsampled and cropped to yield a final resolution of 87×58. After normalization in which each pixel value is scaled to the range from -0.5 to 0.5, the processed depth image serves as the input to the visual encoder network, which operates concurrently with the motion policy at a frequency of 50 Hz. Moreover, owing to the inherent imaging principles of the RealSense D435i, the raw depth images often contain considerable invalid regions and noise. To address these limitations, RGB-Depth Processing module in OpenCV is employed to process the depth map data, effectively enhancing the image quality.

S4. Additional Rewards for BBC

To achieve a smoother and more stable controller, we introduce the following regularization rewards for the BBC:

- Joint torque reward: $r^{\mathrm{T}}_{\mathrm{torque}} = -10^{-5} \cdot \sum ||\tau||^2$ penalizes excessive joint torques.
- Torque change reward: $r^{\mathrm{T}}_{\mathrm{delta\_torque}} = -10^{-7} \cdot \sum ||\Delta_\tau||^2$ penalizes rapid changes in joint torque, where Δ_τ is the difference between consecutive steps.
- Joint acceleration reward: $r^{\mathrm{T}}_{\mathrm{dof\_acc}} = -2.5 \times 10^{-7} \cdot \sum ||\ddot{\mathbf{q}}||^2$ penalizes excessive joint acceleration.
- Joint position limitation reward: $r^{\mathrm{T}}_{\mathrm{joint\_pos\_limit}} = -0.1 \cdot \sum (\max(0, \mathbf{q}_{\min} - \mathbf{q}) + \max(0, \mathbf{q} - \mathbf{q}_{\max}))$ penalizes joint positions that exceed their defined limits.
- Joint velocity limitation reward: $r^{\mathrm{T}}_{\mathrm{joint\_vel\_limit}} = -0.1 \cdot \sum (\min(1, \max(0, |\dot{\mathbf{q}}| - \mathbf{q}_{\mathrm{vel\_limit}})))$ penalizes joint velocities exceeding the velocity limit.
- Joint torque limitation reward: $r^{\mathrm{T}}_{\mathrm{joint\_torque\_limit}} = -0.03 \cdot \sum (\max(0, |\tau| - \tau_{\mathrm{limit}}))$ penalizes joint torques that approach or exceed the torque limit.

- Collision reward: $r^{\mathrm{T}}_{\mathrm{collision}} = -10 \cdot \mathbb{I}_{\mathrm{collision}}$ penalizes collisions involving the thigh and calf.
- Action smoothness reward: $r^{\mathrm{T}}_{\mathrm{delta\_action}} = -0.1 \cdot ||\Delta_a||^2$ penalizes abrupt changes in action commands.

S5. Additional rewards for TSC

To achieve a smoother and safer TSC, we introduce the following regularization rewards:

- Action smoothness reward: $r^{\mathrm{TSC}}_{\mathrm{delta\_action}} = -0.1 \cdot ||\Delta_{a^{\mathrm{T}}}||^2$ penalizes abrupt changes in action commands.
- Collision reward: $r^{\mathrm{TSC}}_{\mathrm{collision}} = -20 \cdot \mathbb{I}_{\mathrm{collision}}$ penalizes collisions involving the thigh and calf.
- Edge contact reward: $r^{\mathrm{TSC}}_{\mathrm{feet\_edge}} = -\sum_{i=0}^{4} c_i^{\mathrm{foot}} M(p_i)$ penalizes each foot contact with the edge of the obstacles. $c^{\mathrm{foot}_i} = 1$ if foot i is in contact. M is a boolean function which is 1 if the foot position p_i within the obstacle edge.

S6. Neural network architectures

The BBC's policy network outputs target positions for the quadrupedal robot's 12 joints. This network is an MLP with 3 hidden layers {512, 256, 128}. The input includes commands, measured and estimated proprioception, and latent physical parameters. Estimated proprioception is generated by a two-layer MLP (sizes {128, 64}), based on measured proprioception. Latent physical parameters are inferred from proprioception history using two 1D convolutional layers with kernel sizes {4, 2} and strides {2, 1}.

The TSC-teacher's policy network outputs both discrete and continuous commands. Discrete commands are probabilities for 5 behaviors, while continuous commands consist of 5×6 values corresponding to the behaviors. The TSC policy network is an MLP with 3 hidden layers {512, 256, 128}. Its input consists of scandots, privileged information, and proprioception. Scandots are an 11×12 matrix, which is flattened and input into a three-layer MLP encoder (sizes {128, 64, 32}), with the 32-dimensional output used as input to the policy network. Estimated proprioception is predicted in the same way as in the BBC.

The TSC-student's policy network is an MLP with 3 hidden layers {512, 256, 128}. It's outputs are identical to the TSC-teacher. Instead of scandots, it uses the current depth map, sized 87×58 as exteroceptive perception. This depth map is processed with 2 convolutional layers (kernel sizes {5, 3}), one max pooling layer (kernel size 2, stride 2), and two fully connected layers to produce a 32-dimensional depth feature. This feature is concatenated with the measured proprioception and passed into a GRU with a 512-unit hidden size to infer the privileged information from historical data.

S7. Human-robot collaboration

In the quadrupedal agility challenge, the robot follows a specific route through randomly placed obstacles. However, in situations where the route is ambiguous, human collaboration is needed to complete the task, similar to the role of handlers in dog

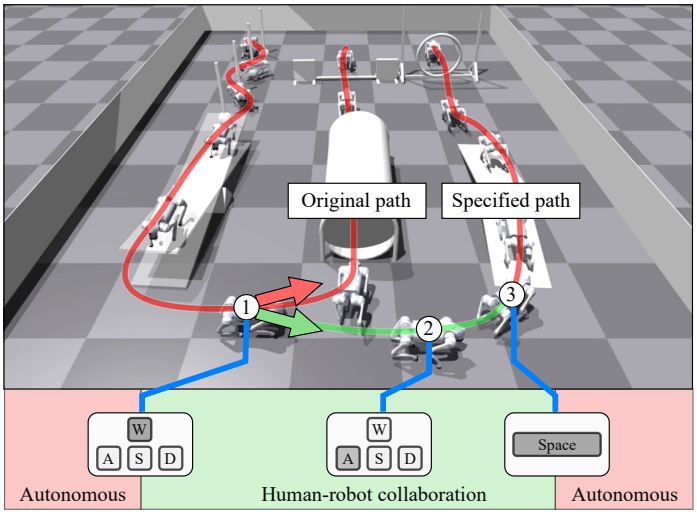

Fig. S1 Human-robot collaboration. The robot receives the operator's input at point 1, where it switches from an immediate left turn to moving straight. The operator only adjusts the linear and angular velocities, while the robot's behavior style remains controlled by the TSC. At point 3, the operator presses the Space key, and the robot transitions from human-robot collaboration to autonomous movement.

agility, where handlers are informed of the route in advance. We simulate this collaborative mode in a scenario with bidirectional routes (Fig. S1). The robot receives the operator's input at point 1, where it switches from an immediate left turn to moving straight. The operator only adjusts the linear and angular velocities, while the robot's behavior style remains controlled by the TSC. At point 3, the operator presses the Space key, and the robot transitions from human-robot collaboration to autonomous movement. We use a remote control for this task, though other methods could also achieve the same goal, such as voice commands, gestures, or behavioral intentions, which we plan to explore in future work. In addition to altering the route, the handler can also modify the skill variable inputs to switch between different behaviors.

S8. Ablation of the BYOL loss for student training

To intuitively demonstrate the improvement in the robustness of the student policy by the BYOL loss, we use Gradient-weighted Class Activation Mapping (Grad-CAM) [64] to visualize the areas of focus in the depth image encoder, as shown in Fig. S2. The first row shows the original depth image containing the bar-jump, as well as the depth images with added random rectangles, white noise, and Gaussian blur. The second row displays the areas of focus for our method's encoder. The third row shows the areas of focus for the encoder without BYOL. With the use of BYOL, the encoder consistently focuses on important areas under various noise disturbances, ensuring the policy's robustness in real-world scenarios.

Table S1 Hyperparameters for semi-supervised InfoGAIL.

Optimizer	Adam
Policy & value learning rate	1.0 E-3
Discriminator learning rate	5.0 E-4
Encoder learning rate	1.0 E-3
Observation horizon	2
Discount factor	0.99
TD(λ)	0.95
PPO clip threshold	0.2
Moving average weight	1.0 E-3
Imitation reward weight	0.2
Task reward weight	0.2

Table S2 Task command ranges for quadrupedal agility challenge.

	Walk	Pace	Trot	Canter	Jump
v_x (m/s)	[0.0, 0.6]	[0.5, 1.5]	[0.5, 1.5]	[0.8, 2.0]	[0.8, 1.8]
v_y (m/s)	[-0.15, 0.15]	[-0.3, 0.3]	[-0.3, 0.3]	[-0.5, 0.5]	[-0.3, 0.3]
ω_z (rad/s)	[-1.0, 1.0]	[-1.57, 1.57]	[-1.57, 1.57]	[-0.5, 0.5]	[-0.5, 0.5]
jump height (m)	-	-	-	-	[0.45, 0.55]
locomotion height (m)	[0.25, 0.34]	[0.25, 0.34]	[0.25, 0.34]	[0.25, 0.34]	-

Table S3 Hyperparameters for EASI.

Optimizer	RMSprop
Discriminator learning rate	3.0 E-4
Discriminator training epoch	10
Number of env.	300
Evolution survival rate	0.5
Trajectory length	500

Table S4 Hyperparameters for privileged learning.

Optimizer	Adam
Policy & value learning rate	5.0 E-4
Discount factor	0.99
TD(λ)	0.95
PPO clip threshold	0.2
Imitation reward weight	0.2
Task reward weight	2.0

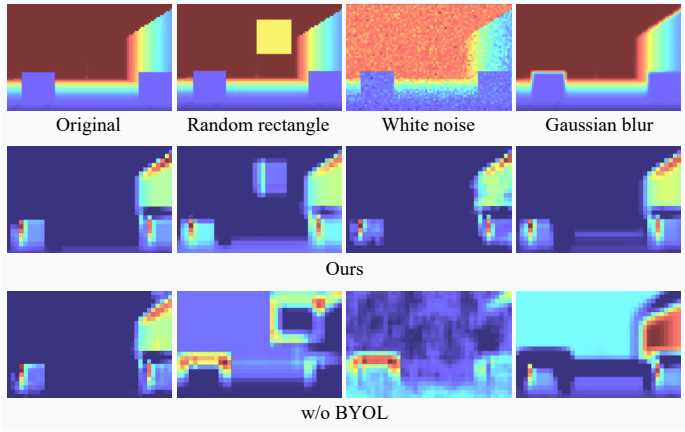

Fig. S2 Visualization of the depth image encoder's focused areas. The first row shows the original depth image, followed by depth images with added random rectangles, white noise, and Gaussian blur. The second row displays the areas of focus for our method's encoder. The third row shows the areas of focus for the encoder without BYOL.

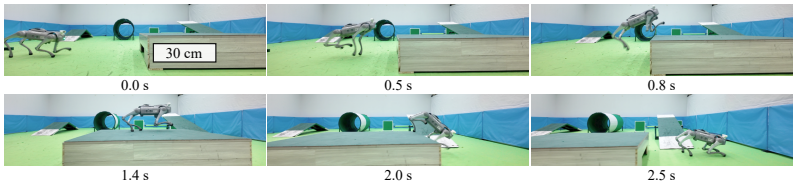

Fig. S3 Box jumping process. The box is 30 cm high, approximately the same as the robot's normal standing height. With our controller, the quadrupedal robot can quickly jump onto the obstacle without relying on its front legs for support.

Algorithm S1 BBC training with semi-supervised InfoGAIL

Initialize policy π_0, value function V_0, discriminator D_0, predictor Q_0, and latent skill distribution $p_0(\mathbf{c})$. Prepare unlabeled and labeled demonstrations d^{E} and d^{EL}
for $i = 0, 1, 2, \cdots N$ **do**
  Sample a batch of latent variables $\mathbf{c} \sim p_i(\mathbf{c})$, $\epsilon \sim p(\epsilon)$
  Interact with the environment using π_i, $\mathbf{c}$ and ϵ, and obtain d^{π}
  Sample a batch of observations $b^{\mathrm{E}} \sim d^{\mathrm{E}}$, $b^{\mathrm{EL}} \sim d^{\mathrm{EL}}$, $b^{\pi} \sim d^{\pi}$
  Update D_i to D_{i+1} and Q_i to Q_{i+1} according to Equation (10)
  Update π_i to π_{i+1} using PPO [58] with reward according to Equation (11)
  Update V_i to V_{i+1} using $TD(\lambda)$
  Update $p_i(\mathbf{c})$ to $p_{i+1}(\mathbf{c})$ with Q_{i+1} and b^{E}
end for

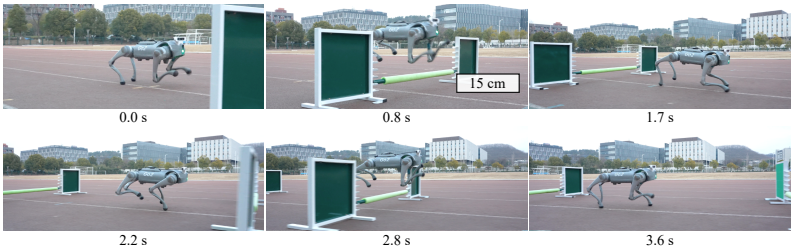

Fig. S4 Hurdling process. We showcase two jumping phases in the hurdling task. The robot utilizes depth vision to naturally switch behaviors, completing the task like a real dog. The whole process can be found in Supplementary Video 3.

Algorithm S2 Simulator parameter optimization with EASI

Initialize parameter distribution $\Xi^{(0)}$, discriminator $\tilde{D}_0$, policy π, real-world demonstration $\tilde{d}^E$
for generation $i = 0, 1, 2, \cdots G$ **do**
  Sample individuals by $\xi_j^{(i)} \sim \Xi^{(i)}$, $j = 1, 2, \cdots, N$
  for simulation environment $j = 1, 2, \cdots, N$ **do**
    Set simulator parameters to $\xi_j^{(i)}$
    Use π to sample trajectory τ_j
    Store τ_j in d^π
  end for
  for update step $k = 1, 2, \cdots, K$ **do**
    Sample a batch of transitions $\tilde{b}^E \sim \tilde{d}^E$, $b^\pi \sim d^\pi$
    Update $\tilde{D}_k$ to $\tilde{D}_{k+1}$ according to Equation (13)
  end for
  Calculate discriminator reward $r^{(i)}$ according to Equation (2)
  Using ES to find next generation distribution $\Xi^{(i+1)} = \mathrm{ES}(\xi^{(i)}, r^{(i)})$
end for

Algorithm S3 TSC training with privileged learning

Initialize teacher policy π_0^{TSC}, student policy $\hat{\pi}_0^{\text{TSC}}$, value function V_0
for $i = 0, 1, 2, \cdots N$ **do**
    Initialize obstacles and robot at a random position
    Interact with the environment using π_i^{TSC}, and obtain d^{π}
    Update π_i^{TSC} to π_{i+1}^{TSC} using Hybrid-PPO [47] with reward according to Equation (14)
    Update V_i to V_{i+1} using $TD(\lambda)$
end for
for $j = 0, 1, 2, \cdots M$ **do**
    Initialize obstacles and robot at a random position
    Interact with the environment using $\hat{\pi}_j^{\text{TSC}}$, and obtain $\hat{d}^{\pi}$
    Update $\hat{\pi}_j^{\text{TSC}}$ to $\hat{\pi}_{j+1}^{\text{TSC}}$ by descending with gradients according to Equation (15)
end for