2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Learning Highly Dynamic Behaviors for Quadrupedal Robots

Chong Zhang[1], Jiapeng Sheng[1], Tingguang Li[1], He Zhang[1],
Cheng Zhou[1], Qingxu Zhu[1], Rui Zhao[1], Yizheng Zhang[1], Lei Han[1]

Abstract— **Learning highly dynamic behaviors for robots has been a longstanding challenge. Traditional approaches have demonstrated robust locomotion, but the exhibited behaviors lack diversity and agility. They employ approximate models, which lead to compromises in performance. Data-driven approaches have been shown to reproduce agile behaviors of animals, but typically have not been able to learn highly dynamic behaviors. In this paper, we propose a learning-based approach to enable robots to learn highly dynamic behaviors from animal motion data. The learned controller is deployed on a quadrupedal robot and the results show that the controller is able to reproduce highly dynamic behaviors including sprinting, jumping and sharp turning. Various behaviors can be activated through human interaction using a stick with markers attached to it. Based on the motion pattern of the stick, the robot exhibits walking, running, sitting and jumping, much like the way humans interact with a pet.**

I. INTRODUCTION

Animals exhibit remarkable agility in the wild. In order to escape from predators, they perform various behaviors such as sprinting, jumping and high-speed sharp turning. Enabling robots to reproduce these behaviors has been a long-standing challenge. As robots move at high speed, several physical considerations begin to influence the dynamics of the robot, including the enforcement of motor limit, the regulation of contact force and posture balance during flight phases [1].

Manually designing controllers to resolve these issues typically requires a lengthy design process and tedious parameter tuning [2], [3], [4]. Conventional approaches break the system into manageable submodules and each submodule generates reference values for the next [5], [6]. Animal data is utilized in [7], [8] to allow the robot to reproduce relatively agile behaviors. However, these methods employ approximate models for each module and lead to compromises in performance. Controllers based on trajectory optimization suffer from expensive computational costs due to the optimization needs to be performed at execution time [9], [10], [11], [12].

Learning-based approaches, such as Reinforcement Learning (RL), promise the potential to overcome the limitations [13], [14], [15], [16]. The work in [1] has shown a neural network controller can push a small quadruped to the limits of its agility, achieving high-speed mobility. However, the task reward function they used is linear and angular velocity tracking, thus the learned motion has limited diversity and is not guaranteed to exhibit animal-like behaviors. In [17], the authors proposed an imitation learning system that enables the quadruped to learn agile locomotion skills by imitating real animals. However, their model can only track low-speed reference trajectories and the robot cannot be guided in a controllable way. Primitive skill reuse has been first investigated in character animation [18], [19], [20] and then expanded to robotics [21], [22]. These approaches compress the primitive motion to a continuous latent space, and navigate this space to accomplish use-specific tasks. In [22], the researchers investigated the use of prior knowledge of animal behaviors to learn reusable locomotive skills in quadruped robots. However, the behaviors performed by their robot also lacks agility and diversity. In [23], the researchers proposed a learning-based approach to learn traversing challenging terrains by imitating animals. The robot can perform natural-looking behaviors on different terrains like stairs and slopes, but in relatively low-speed mode.

In this paper, we propose a data-driven approach that enables the robot to perform various highly dynamic behaviors robustly and human can interact with the robot by actively triggering these skills. Our model is trained in simulation and deployed on a quadrupedal platform designed by our team [3], [4]. The main contribution of our paper is as follows. First, we propose a learning framework that allows the robot to imitate a versatile of motions collected from real animals. The framework utilizes a vector quantized controller [24], [25] and could robustly track high-speed motion data including sprinting, jumping and sharp turning. Second, various behaviors can be activated, allowing humans to interact with the robot in a controllable manner. Humans can utilize a stick with markers attached to it to interact with the robot, much like the way humans interact with a pet. Third, several generic techniques have been proposed to improve the tracking accuracy and robustness of the robot. The tracking accuracy of high-speed motion is improved by utilizing prioritized sampling and early termination. Prioritized sampling is enabled to encourage the model to exploit more highly dynamic motions during training. Early termination provides another means of discouraging undesired behaviors. The robustness of our model deployed on the real robot is achieved by modifying the action space commonly used in previous work [13], [26]. The action space is represented as the change of joint angular velocities in our model, compared with previous approaches the new representation achieves more fluid and robust motion.

[1]Chong Zhang, Jiapeng Sheng, Tingguang Li, He Zhang, Cheng Zhou, Qingxu Zhu, Rui Zhao, Yizheng Zhang and Lei Han are with Tencent Robotics X Laboratory, Shenzhen, Guangdong, China. chongzzhang@tencent.com

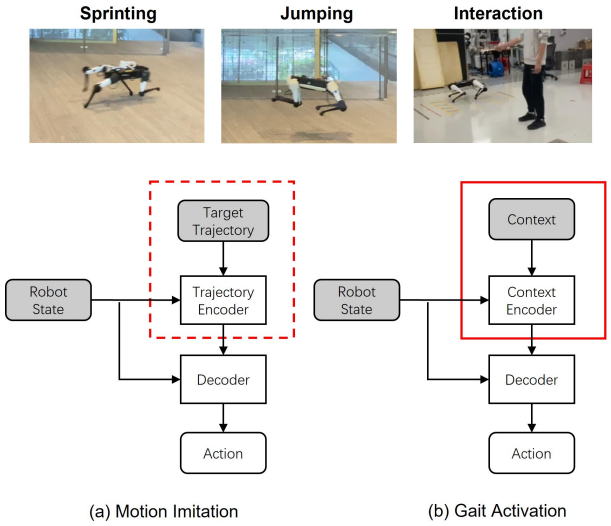

Fig. 1. The framework of proposed model. It consists of two stages: (a) motion imitation and (b) gait activation. In motion imitation stage, the model is trying to imitate animal behaviors. In gait activation stage, humans can interact with the robot by triggering various gaits. The module in red dashed rectangle in (a) is replaced by module in red solid rectangle in (b), the remaining module is shared by (a) and (b).

II. METHODOLOGY

The general architecture of our model is shown in Fig. 1. The model is learned in two stages. In the first stage, we use a trajectory encoder and decoder to imitate the animal motion data. The trajectory encoder encodes the target trajectory as a latent control signal, the decoder outputs action based on the current robot state and the latent control signal. In the second stage, we freeze the parameters of the decoder and replace the trajectory encoder by a context encoder, which tries to learn appropriate latent control signals to navigate the robot based on desired commands.

A. Data Acquisition and Processing

The training data is collected using motion capture system. We use a medium-sized Labrador retriever as the subject and capture locomotive data of various gaits. The data are saved according to gait type. The subject is guided by a dog trainer to follow various instructions. The gaits include standing, walking, running jumping and sitting. Each gait is repeated 4-6 times to ensure data diversity. The moving trajectories consist of square, circle and straight line. Upon completion, we have approximately 0.5 hour of motion data. We manually label the gait at frame level and plot the statistics in Fig. 2. Horizontal axis represents gait type, vertical axis represents the duration in minutes. Since the morphology of real animal is different from the robot, we retarget the source motion to the robot using inverse kinematics [27].

B. Motion Imitation

The motion imitation module consists of a trajectory encoder, a decoder and a latent embedding space. The encoder and decoder are parameterized using neural networks and the embedding space is a K-way categorical discrete space,

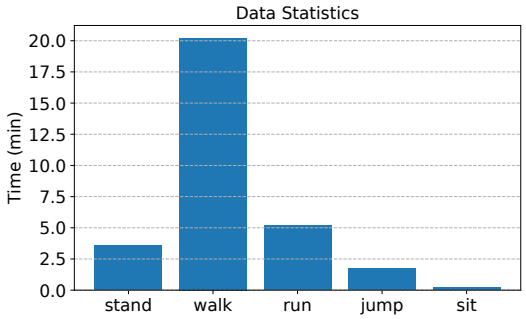

Fig. 2. Statistics of collected motion data from a medium-sized Labrador retriever. Horizontal axis represents gait type, vertical axis represents the duration in minutes.

where K is the size of the space. The benefits of using discrete over continuous latent space has been shown to improve exploration efficiency for downstream tasks [25]. The input to the encoder includes current proprioception of the robot and target trajectories within one second time window in the future. The encoder encodes the current robot state and target future state as a control signal in latent space. The current proprioception of the robot and the latent control signal serve as the input to the decoder. The output of the decoder is processed and fed into a proportional-derivative (PD) controller. The PD controller then computes the joint torque for each actuator. Compared with direct output joint torques, PD controllers have been shown to improve performance and learning speed for motor control tasks [28].

The proprioception of the robot $\mathbf{s}_t^{\mathrm{p}}$ at timestep t is the concatenation of three consecutive states spanning from timestep *t-2* to t and history actions from timestep *t-3* to *t-1*. The state at each timestep consists of joint angles, joint angular velocities, root angular velocities and gravity vector. The gravity vector is used to represents the roll and pitch of the robot [29]. Root angular velocities are expressed under the robot coordinate system. The history action is the previous output from the decoder. The future trajectories $\mathbf{s}_t^{\mathrm{f}}$ at timestep t specifies the target pose of the robot. It consists of states of reference motion located 0.03s, 0.06s, 0.3s and 1s relative to current timestep. Each reference state includes target joint angles, target root position and orientation expressed under the robot coordinate system. The input to the encoder is the concatenation of $\mathbf{s}_t^{\mathrm{p}}$ and $\mathbf{s}_t^{\mathrm{f}}$, the output of the encoder $\mathbf{z}_t$ is mapped to the nearest element of the embedding space as $\mathbf{c}_{t,k}$, where $\mathbf{c}_{t,k}$ is the k-th element in the embedding space and $k = \arg\min_j ||\mathbf{z}_t - \mathbf{c}_{t,j}||_2^2$. The decoder $\pi(\mathbf{a}_t|\mathbf{s}_t^{\mathrm{p}}, \mathbf{c}_{t,k})$ is conditioned on the proprioception and the embedding vector $\mathbf{c}_{t,k}$ and outputs the distribution of action $\mathbf{a}_t$.

We try two types of action space and compare the tracking performance under highly dynamic condition. The first type of action space $\mathbf{a}_t^{\mathrm{I}}$ represents the change of joint angles and is obtained from the output of the decoder. The target joint angles for the PD controller $\mathbf{q}_t^*$ are computed as:

$$\mathbf{q}_t^* = \mathbf{q}_t + \mathbf{a}_t^{\mathrm{I}} \tag{1}$$

where $\mathbf{q}_t$ is the current joint angles of the robot. The target

joint angular velocities $\dot{\mathbf{q}}_t^*$ are set to zero [13], [26].

The second type of action space a_t^{II} represents the change of joint angular velocities. The target angular joint velocities and the target joint angles are computed as:

$$\dot{\mathbf{q}}_t^* = \dot{\mathbf{q}}_t + \mathbf{a}_t^{\text{II}} \tag{2}$$

$$\mathbf{q}_t^* = \mathbf{q}_t + \dot{\mathbf{q}}_t^* \Delta t \tag{3}$$

where Δt is the timestep. The current joint angles, angular velocities of the robot, target joint angles and the target joint angular velocities are fed to the PD controller.

The reward for motion tracking task is defined as:

$$r_t^{\text{tr}} = r_t^{\text{i}} + \alpha r_t^{\text{q}} \tag{4}$$

where r_t^{i} is the imitation reward, r_t^{q} is the vector quantized reward and α is the weighting coefficient. The imitation reward is the weighted sum of pose reward, velocity reward, end-effector reward, root pose reward and root velocity reward. It is defined the same as in [17]. The vector quantized reward is define as [30]:

$$r_t^{\text{q}} = -\|\text{sg}[\mathbf{z}_t] - \mathbf{c}_{t,*}\|_2^2 - \beta\|\mathbf{z}_t - \text{sg}[\mathbf{c}_{t,*}]\|_2^2 \tag{5}$$

where sg is the stopgradient operator, $\mathbf{c}_{t,*}$ is the selected embedding vector and β is a coefficient. The first term tries to move the embedding vectors towards the encoder output, the second term is used to make sure the encoder output commit to the embedding vectors. We use proximal-policy optimization (PPO) to update the model parameters [31].

Imitating highly dynamic behaviors such as jumping is challenging. We use prioritized sampling (PS) and early termination (ET) to facilitate motion imitation. Prioritized sampling is enabled to encourage the model to exploit more data with low tracking accuracy. During training, at the beginning of each episode, a particular motion data is selected according to a performance score. The score is computed as the inverse of normalized cumulative imitation reward. The early termination provides another means of reward shaping to discourage undesired behaviors [13]. We find that setting obstacles under the flight trajectories encourages the learned policy to achieve consistent peak height during flight phase with animal data. During training, the peak heights of the retargeted jumping data are detected and obstacles are placed with proper pose. The shape of the obstacle is a cube with length, width and height equal to 0.5m, 0.02m and 0.2m respectively. The size of the cube is chosen to avoid collision with the robot during jumping. The obstacle position is computed by projecting the animal's position at peak height on the ground and the orientation is set equal to the animal's orientation at peak height. Early termination is triggered when detection of a collision, characterized by the robot's body or legs making contact with the cube.

C. Gait Activation

Gait activation is achieved by removing the trajectory encoder and stacking another context encoder on top of the decoder. The inputs to the context encoder include the current proprioception $\mathbf{s}_t^{\text{p}}$ and context state $\mathbf{s}_t^{\text{c}}$. The context

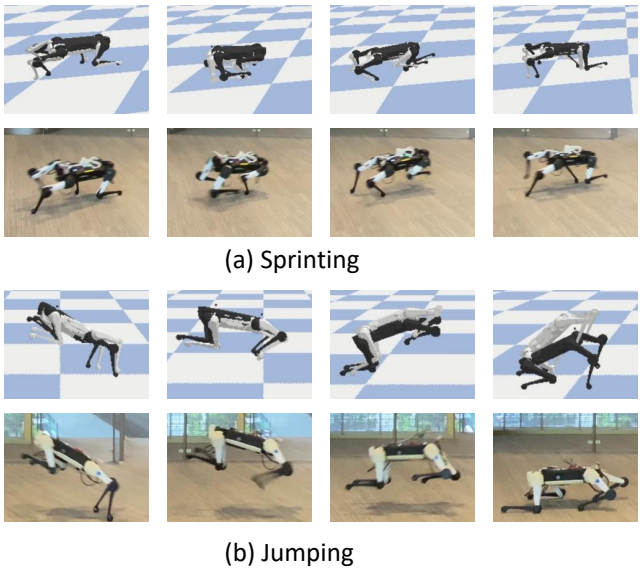

Fig. 3. Snapshots of learned robot behaviors. (a) Snapshots of sprinting. (b) Snapshots of jumping

state consists of current linear velocity, current root height of the robot, target linear velocity and target height. During training, target direction is uniformly sampled between 0 and 2π, target speed in horizontal plane is uniformly sampled between 0 and 3m/s. Target height is uniformly sampled from three values 0.1m (sit), 0.3m (walk and run) and 0.5m (jump). The linear velocity and root height of the robot are obtained using a motion capture system when deployed on real system. The output of the encoder is a categorical distribution that is used to sample the appropriate embedding vector from the discrete embedding space learned in previous section. The selected embedding $\mathbf{c}_{t,*}$ and $\mathbf{s}_t^{\text{p}}$ are fed to the decoder.

The context encoder is parameterized with a neural network and is updated using PPO. The reward is defined as:

$$r_t^{\text{at}} = r_t^{\text{c}} + \gamma r_t^{\text{g}} \tag{6}$$

where r_t^{c} is the context reward to align the robot liner velocity and root height with the desired velocity and height and is computed similar with [19], r_t^{g} is a generative adversarial reward that is used to encourage the produced motion similar to motions learned from animal data [32] and γ is the coefficient. The context reward is defined as:

$$r_t^{\text{c}} = e^{\cos(\theta_t - \theta_t^{\text{d}}) - 1} \times e^{-|v_t - v_t^{\text{d}}|} \times e^{-|h_t - h_t^{\text{d}}|} \tag{7}$$

where θ_t is the current facing direction of the robot projected on horizontal plane, θ_t^{d} is the desired facing direction, v_t is the speed of the robot in horizontal plane, v_t^{d} is the desired speed in horizontal plane, h_t is the root height of the robot and h_t^{d} is the desired height. The imitation reward is defined as:

$$r_t^{\text{g}} = -\log(1 - D(\mathbf{s}_t^{\text{p}}, \mathbf{a}_t)) \tag{8}$$

where $D(\mathbf{s}_t^{\text{p}}, \mathbf{a}_t)$ is a discriminator to classify the state-action pair is produced from gait activation stage or motion

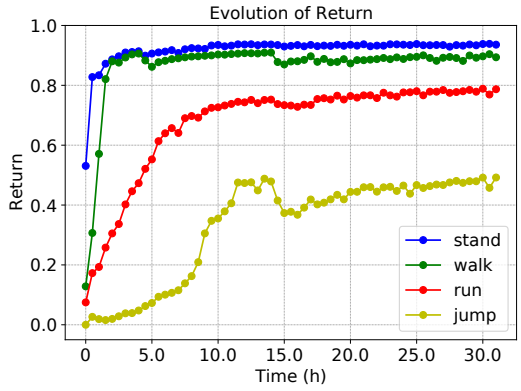

Fig. 4. The evolution of imitation performance. Horizontal axis represents training time in hours, vertical axis represents average return. Colors represent different gaits.

imitation stage. We use the same objective function for the discriminator as in [33]. Compared to directly learning gait activation with the imitation reward, our two phase learning framework is able to reuse the the skills learned in the first stage for various downstream tasks.

D. Real World Transfer

Our model is trained in simulation and deployed on a quadrupedal platform. To facilitate the transfer of learned policy from simulation to real world, we follow most existing approaches proposed by [13], [26], [34]. We randomize the terrain friction, the actuators' torque limit, and periodically apply random forces on the root of the robot. At the beginning of each episode, the lateral friction of the terrain is uniformly sampled between 0.9 to 2.0, the torque limit of each actuator is uniformly sampled between 16 to 20 $N \cdot m$. The magnitude of the random force is uniformly sampled between 0 to 10 N and the direction is uniformly sampled from a spherical surface. The force is periodically applied and last for 0.2s each time.

III. EXPERIMENTAL RESULTS

We use a quadrupedal platform independently designed by our team [3]. The quadruped consists of twelve actuators with three (hip, upper leg and knee) on each leg. Pybullet is used as the simulation environment during training [35]. A distributed reinforcement learning infrastructure named TLeague [36] is employed to increase training efficacy. We first show the results by enabling all the techniques we proposed and then compare the full method with alternatives that disable some components. We find that prioritized sampling and early termination are helpful for tracking the jumping trajectories and action space with change of joint angular velocity help the system to achieve robust tracking of the sprinting trajectories.

A. Motion Imitation

The snapshots of learned behaviors are shown in Fig. 3. Simulation results are shown on top of real robot behaviors. In the simulation, the white robot is the kinematic reference data and the black robot is the physics-based animation.

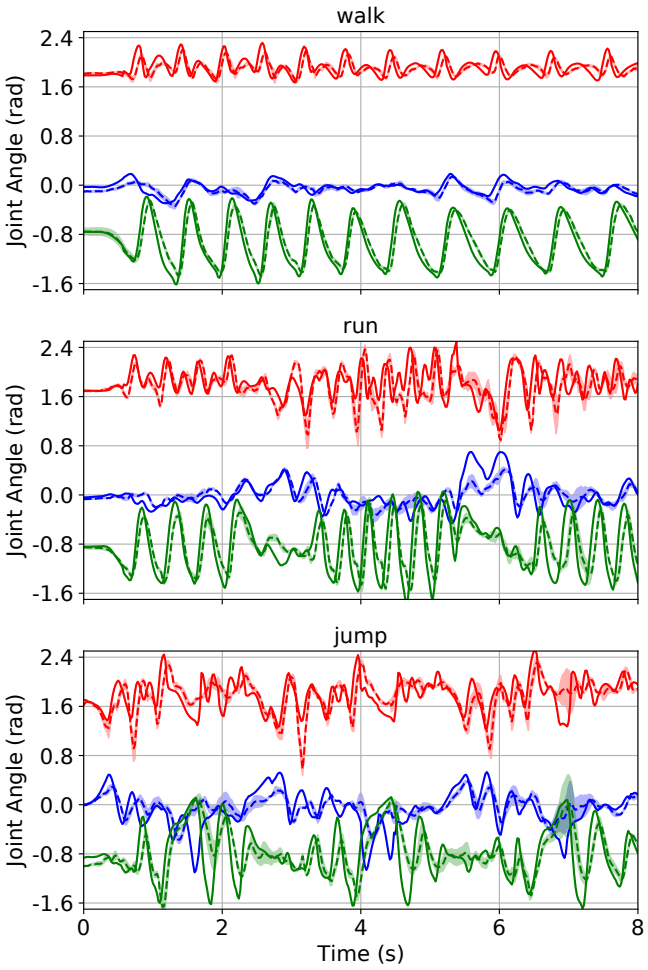

Fig. 5. Joint angle curves tested at the end of training. Only curves from front left leg are shown for clarity. Top, mid and bottom panel represent curves of representative walking, running and jumping motion. Horizontal axis represents time in second and vertical axis represent joint angle in radian. Solid lines represents reference data, dashed lines represent the joint angles of real robot. Shaded areas represents the standard deviation computed over three trials. Hip joint, upper leg joint and knee joint are colored with red, blue and green respectively.

We can see that the robot is able to accurately imitate the reference data most of the time except during the falling stage of the jumping action.

Fig. 4 shows the evolution of imitation performance obtained in simulation. During training, model parameters are saved at checkpoints and the performance is measured by the average reward over episodes and timesteps. Horizontal axis represents time in hours and vertical axis represents reward tested at corresponding checkouts. Different colors correspond to different gaits. We can see that standing and walking learn faster compared to high dynamic motion (running and jumping). The final performance of jumping is relatively low, it may be due to the physical constraints of the robot, such as the enforcement of motor limit and the misalignment of the body configuration between the quadrupedal robot and the real animal.

Fig. 5 shows the joint angle curves at the end of training tested on the real robot. We test the performance of walking, sprinting and jumping and show representative curves of

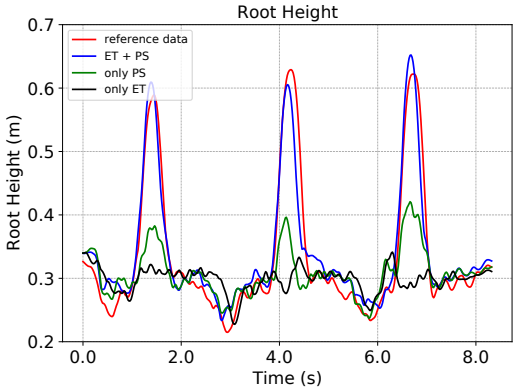

Fig. 6. Jumping performance measured at the end of training. One representative motion data is used. The data lasts around eight seconds and includes three jumping actions. The performance is measured by the root height. Red curve represents reference data, blue curve is obtained by enabling prioritized sampling (PS) and early termination (ET), green and black curves are obtained by enabling either PS or ET.

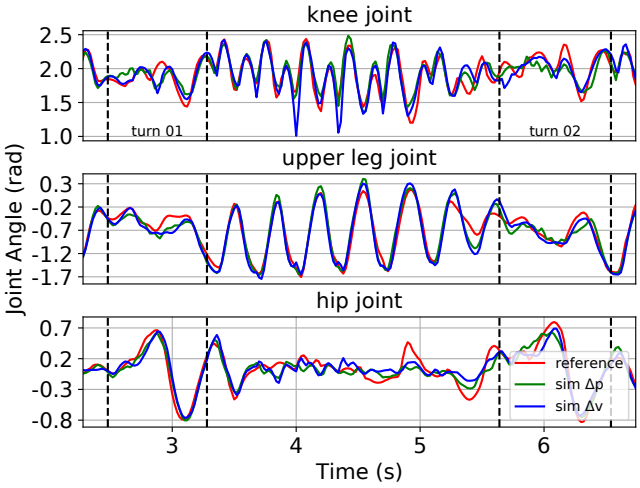

Fig. 7. Joint angle curves of sprinting motion tested under different action space representations in simulation. Only curves of front left leg are shown for clarity. Knee joint, upper leg joint and hip joint are shown from top to bottom. Red curves are reference motion, green curves are obtained by action space representation with change of joint angle, denoted as Δp, blue curves are obtained by action space representation with change of angular velocity, denoted as Δv.

each gait. Each panel in the figure represents the curves from front left leg, solid lines is the motion of reference data and dashed lines are from joint encoders of the real robot. Shaded areas are the standard deviations averaged over three trials. The curves of the other legs are similar with a slight difference between front and back legs. Hip joint, upper leg joint and knee joint are colored with red, blue and green respectively. Tracking of the joint angle is relatively accurate for low speed motion except with a phase shift. The phase shift may due to the communication delay of the real robotic system. As the speed of the motion becomes higher, the tracking performance gets worse, the trend is also reflected in the learning curve in Fig. 4. Factors that may impact the performance include the discrepancy of the actuator model between the simulation and real robot and the simplified contact model used in simulation.

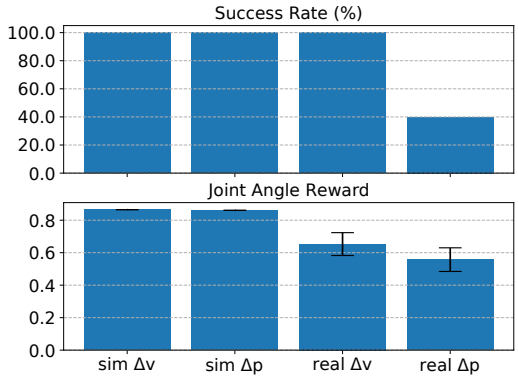

Fig. 8. Performance of imitating sprint data under different conditions. The results are obtained both in simulation and real robot platform. Top panel shows the success rate and bottom panel shows the joint angle tracking accuracy. The arrow bars in bottom panel from real robot experiments are obtained from five trials.

To evaluate the impact of prioritized sampling (PS) and early termination (ET) on the learned highly dynamic behaviors, we compare the full method with alternatives that disable some components. Fig. 6 plots the jumping performance of one representative motion data at the end of training tested in simulation. The tested data lasts around eight seconds and consists of three jumping actions. The performance is measured by the root height. The red curve represents the root height of reference data. The blue curve is probed from the model learned by enabling PS and ET. The green curve and black curves are obtained by enabling either PS or ET. We can see that enabling PS and ET allows the model to achieve consistent peak height with the reference data.

Fig. 7 plots joint angle curves of sprinting motion tested under different action space representations in simulation. We select a representative sprinting data which consists of three straight runs and two sharp turnings. The peak base linear speed is around 4 m/s. Only curves of front left leg are shown for clarity. Knee joint, upper leg joint and hip joint are shown from top to bottom. Red curves are reference motion, green curves are obtained by action space representation with change of joint angle, denoted as Δp, blue curves are obtained by action space representation with change of joint angular velocity, denoted as Δv. The turning periods are shown between two black vertical dashed lines. We notice oscillations of front leg during turning motion when first trying Δp representation, as reflected from the green knee joint curve. This representation is commonly used in previous work [13], [26] under low-speed motion and always accompanied with additional joint smooth term added to reward. The robot exhibits smoother motion when using Δv representation, as reflected from the blue joint curves.

Fig. 8 compares the performance of imitating sprinting data under two action space representations. The results are obtained both in simulation and real robot platform. Top panel shows the success rate and bottom panel plots the joint angle tracking accuracy. We can see that in simulation, the two action space representation exhibit similar performance, while in real robot platform, the action space with change

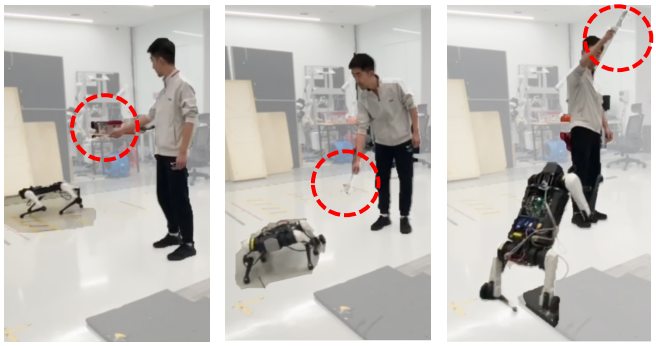

Fig. 9. Snapshots of gait activation. Left: walk, middle: sit, right: jump. The stick with markers are labeled within the red dashed circle.

of joint angular velocity is superior than action space with change of joint angle. We repeat the experiment of tracking sprinting data five times and compute the success rate and joint angle reward. A successful trail is defined as completing the tracking without any falling. The most challenging period is the speed up phase after the sharp turning, the failure occurs during this period most of the time. The action space with change of joint angular velocity achieves robust tracking with higher accuracy.

B. Gait Activation

Fig. 9 shows snapshots of gait activation deployed on real robot. Humans can interact with the robot using a stick with markers attached to it. The robot follows the velocity of the stick in horizontal plane and the position of the stick in vertical axis and exhibits various behaviors. The motion of the stick is perceived using a motion capture system. Walking, sitting and jumping are shown from left to right. The stick with markers are labeled within the red dashed circle. The full video can be seen in the supplementary materials.

We compare the velocity tracking performance under two action space representations in simulation and Fig. 10 shows the results. We vary target direction and target speed and compute the velocity tracking reward as in [19]. Target speed is varied within range 0.5-3m/s with spacing 0.5m/s. Target direction in horizontal plane is varied within range 0-360 degrees with spacing 22.5 degrees. The initial orientation of the robot is aligned with the target direction. Fig. 10 is shown in polar coordinate, angle represents target direction, radius represents target speed and density represents the tracking reward. Top panel is the results obtained using action space with change of joint angular velocity, bottom panel is obtained using action space with change of joint angle. We can see that the overall performance of the top panel is better than bottom panel. In the bottom panel, as the target speed increase, the velocity tracking reward decrease, while in the top panel the performance is relatively robust to target speed changes.

IV. CONCLUSIONS

In this paper, we propose a learning-based approach to enable robot to perform various highly dynamic skills and

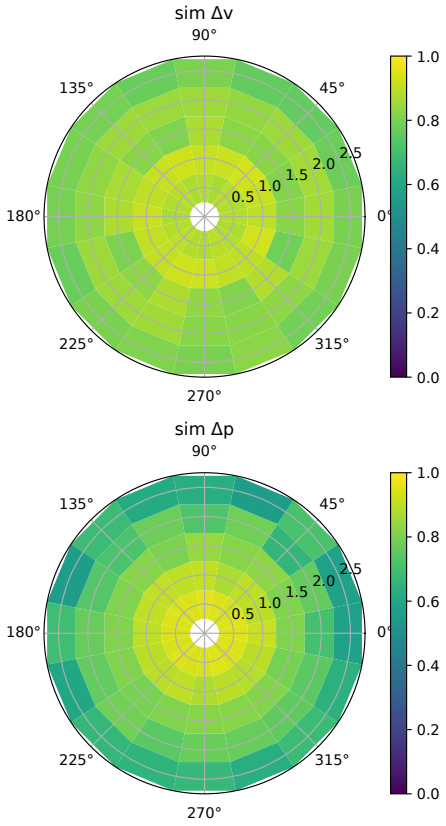

Fig. 10. Polar plot of velocity following performance. Angle represents target direction, radius represents target speed and density represents the tracking reward. Top panel is the results obtained using action space with change of joint angular velocity, bottom panel is obtained using action space with change of joint angle.

these skills can be activated through human interaction much the same way humans interact with a pet. The proposed approach can be robustly deployed on the real quadrupedal robot. One future direction could consider active compliance control when the robot performs highly dynamic behaviors. This would help to minimize the contact force induced by landing after jump motion or speeding up followed by sharp turning. Another possible extension of this work could be to enable vision-based perception, which would allow humans to interact with the robot with more flexibility.

APPENDIX

The trajectory encoder and context encoder are both a multilayer perception (MLP) with 2 hidden layers, each has 256 hidden units and ReLU activation function. The output is a linear layer and has dimension of 32, which corresponds to the dimension of the embedding space. The embedding space is a categorical discrete space with dictionary size of K=256. The decoder is also a MLP with 2 hidden layers, each has 256 hidden units and ReLU activation function. The output is a linear layer with dimension of 12, which corresponds to the number of actuators of the robot. The discriminator used in equation (8) is a MLP with 2 hidden layers and 256 hidden units each layer. The weighting coefficients in the rewards are set as follows: α=1, β=0.25, γ=1.

REFERENCES

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

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

[3] W. Chi, X. Jiang, and Y. Zheng, "A linearization of centroidal dynamics for the model-predictive control of quadruped robots," in *2022 IEEE International Conference on Robotics and Automation*. IEEE, 2022, pp. 4656–4663.

[4] Q. Zhou, S. Yang, X. Jiang, D. Zhang, W. Chi, K. Chen, S. Zhang, J. Li, J. Zhang, R. Wang *et al.*, "Max: A wheeled-legged quadruped robot for multimodal agile locomotion," *IEEE Transactions on Automation Science and Engineering*, 2023.

[5] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 811–818.

[6] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2665–2670.

[7] D. Kang, S. Zimmermann, and S. Coros, "Animal gaits on quadrupedal robots using motion matching and model-based control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8500–8507.

[8] D. Kang, F. De Vincenti, N. C. Adami, and S. Coros, "Animal motions on legged robots using nonlinear model predictive control," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 955–11 962.

[9] C. Gehring, S. Coros, M. Hutter, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger *et al.*, "Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot," *IEEE Robotics & Automation Magazine*, vol. 23, no. 1, pp. 34–43, 2016.

[10] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory optimization through contacts and automatic gait discovery for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1502–1509, 2017.

[11] J. Carius, R. Ranftl, V. Koltun, and M. Hutter, "Trajectory optimization with implicit hard contacts," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3316–3323, 2018.

[12] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[13] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[14] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions. 2022 ieee," in *RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, 2022.

[15] Y. Ji, Z. Li, Y. Sun, X. B. Peng, S. Levine, G. Berseth, and K. Sreenath, "Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1479–1486.

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

[17] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[18] S. Levine, J. M. Wang, A. Haraux, Z. Popović, and V. Koltun, "Continuous character control with low-dimensional embeddings," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–10, 2012.

[19] H. Y. Ling, F. Zinno, G. Cheng, and M. Van De Panne, "Character controllers using motion vaes," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 40–1, 2020.

[20] N. Wagener, A. Kolobov, F. Vieira Frujeri, R. Loynd, C.-A. Cheng, and M. Hausknecht, "Mocapact: A multi-task dataset for simulated humanoid control," *Advances in Neural Information Processing Systems*, vol. 35, pp. 35 418–35 431, 2022.

[21] X. Huang, Z. Li, Y. Xiang, Y. Ni, Y. Chi, Y. Li, L. Yang, X. B. Peng, and K. Sreenath, "Creating a dynamic quadrupedal robotic goalkeeper with reinforcement learning," *arXiv preprint arXiv:2210.04435*, 2022.

[22] S. Bohez, S. Tunyasuvunakool, P. Brakel, F. Sadeghi, L. Hasenclever, Y. Tassa, E. Parisotto, J. Humplik, T. Haarnoja, R. Hafner *et al.*, "Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors," *arXiv preprint arXiv:2203.17138*, 2022.

[23] T. Li, Y. Zhang, C. Zhang, Q. Zhu, W. Chi, C. Zhou, and L. Han, "Learning terrain-adaptive locomotion with agile behaviors by imitating animals," *arXiv preprint arXiv:2308.03273*, 2023.

[24] L. Han, Q. Zhu, J. Sheng, C. Zhang, T. Li, Y. Zhang, H. Zhang, Y. Liu, C. Zhou, R. Zhao *et al.*, "Lifelike agility and play on quadrupedal robots using reinforcement learning and generative pre-trained models," *arXiv preprint arXiv:2308.15143*, 2023.

[25] Q. Zhu, H. Zhang, M. Lan, and L. Han, "Neural categorical priors for physics-based character control," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 6, pp. 1–16, 2023.

[26] 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.

[27] M. Gleicher, "Retargetting motion to new characters," in *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, 1998, pp. 33–42.

[28] X. B. Peng and M. Van De Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?" in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2017, pp. 1–13.

[29] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020.

[30] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

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

[32] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

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

[34] 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.

[35] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[36] P. Sun, J. Xiong, L. Han, X. Sun, S. Li, J. Xu, M. Fang, and Z. Zhang, "Tleague: A framework for competitive self-play based distributed multi-agent reinforcement learning," *arXiv preprint arXiv:2011.12895*, 2020.