Learning Steerable Imitation Controllers from Unstructured Animal Motions

Dongho Kang*, Jin Cheng*, Fatemeh Zargarbashi*, Taerim Yoon†, Sungjoon Choi†, and Stelian Coros*

***Abstract*— This paper presents a control framework for legged robots that leverages unstructured real-world animal motion data to generate animal-like and user-steerable behaviors. Our framework learns to follow velocity commands while reproducing the diverse gait patterns in the original dataset. To begin with, animal motion data is transformed into a robot-compatible database using constrained inverse kinematics and model predictive control, bridging the morphological and physical gap between the animal and the robot. Subsequently, a variational autoencoder-based motion synthesis module captures the diverse locomotion patterns in the motion database and generates smooth transitions between them in response to velocity commands. The resulting kinematic motions serve as references for a reinforcement learning-based feedback controller deployed on physical robots. We show that this approach enables a quadruped robot to adaptively switch gaits and accurately track user velocity commands while maintaining the stylistic coherence of the motion data. Additionally, we provide component-wise evaluations to analyze the system's behavior in depth and demonstrate the efficacy of our method for more accurate and reliable motion imitation.**

I. Introduction

Motion imitation using reinforcement learning (RL) leverages prerecorded motion data from real-world animals or human actors, offering an efficient way to acquire natural and agile locomotion skills for legged robots [1, 2, 3]. By directly imitating motion data, this approach eliminates the need for hand-crafted reference generators or heuristic reward designs commonly used in RL training.

However, prerecorded motion data consists of fixed motion trajectories that cannot be modified during execution, preventing any real-time user steering or interaction. As a result, the learned skills are typically limited to specific scenarios that closely mirror the original recording conditions. Furthermore, significant differences in morphology and physical properties between the data source and the target robot often make it difficult to reproduce the motions effectively.

To overcome these limitations, this paper presents a framework for steerable animal motion imitation in legged robots by leveraging the diverse movement patterns present in an unstructured motion database (DB). Here, *steerable* refers to the ability to responsively follow a user's intended moving direction and velocity, and adapt gait modes based on these commands. The term *unstructured* indicates that our approach utilizes a rich dataset contains a wide range of gait patterns spanning various movement speeds, notably without being pre-segmented by gait pattern and velocity.

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 866480.)

*The authors are with the Computational Robotics Lab in the Department of Computer Science, ETH Zurich, Switzerland. {kangd, jicheng, fzargarbashi, scoros}@ethz.ch

†The authors are with the Robot Intelligence Lab in the Department of Artificial Intelligence, Korea University, South Korea. {taerimyoon, sungjoon-choi}@korea.ac.kr

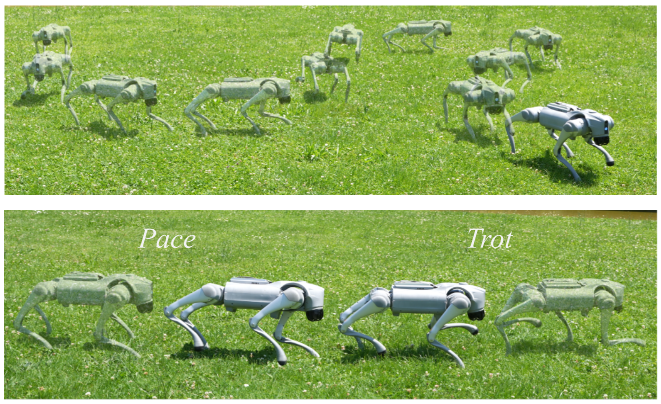


Fig. 1: *Unitree Go2* robot navigating freely across a grass field in response to joystick commands (**top**). The gait pattern automatically transitions from *Pace* to *Trot* as the forward speed command increases from $0.6\,\mathrm{m/s}$ to $1.0\,\mathrm{m/s}$ (**bottom**).

The framework begins with an offline kino-dynamic motion retargeting process that adapts the motion data to the robot's morphology and physical capabilities, mitigating kinematic and physical artifacts. To generate responsive behaviors, we employ a variational autoencoder (VAE)-based motion synthesis module [4] that generates reference robot motions conditioned on user-specified velocity commands, capturing the diverse gait patterns in the motion data. These reference motions are executed by a feedback control policy, trained via RL with the motion synthesis module in the loop.

We validate the efficacy of our approach through comprehensive experiments evaluating the core stages of the framework. Our kino-dynamic motion retargeting method effectively generates kinematically and dynamically feasible motions, enabling reliable RL training that accurately replicates dynamic and expressive animal motions. By effectively reproducing diversity in the motion database, the motion synthesis module generates reference trajectories that not only track user-specified velocity commands accurately but also adaptively switch between gait patterns. Finally, we demonstrated the full pipeline by deploying motion synthesis and control online on the *Unitree Go2* robot, showcasing responsive navigation and adaptive gait switching behavior in real time, as shown in Figure 1.

In summary, this paper makes three main contributions: first, we introduce a framework that reproduces diverse behaviors from unstructured motion data on legged robots; second, we present a kino-dynamic motion retargeting strategy that bridges the morphological and physical gaps for seamless skill transfer; and finally, we demonstration of steerable legged locomotion control with natural gait transitions on a quadruped robot while replicating animal motion style.

II. Related Work

A. Motion Imitation for Legged Robots

In pursuit of replicating the natural and agile movements of legged animals and humans, an increasing number of studies leverage motion data captured from real subjects, either through 3D motion capture systems [1, 3, 5, 6, 7, 8] or pose estimation from monocular video [2, 3, 9].

A key challenge in this direction is bridging the morphological and physical gap between the motion source and the target robot—a process known as *motion retargeting* (MR). Earlier efforts primarily addressed kinematic discrepancies by using scaled keypoint transfers [10], or remapping high-level motion features such as contact timings and base trajectories [5, 11]. However, when the source motion exhibits complex and highly dynamic behaviors, difference in physical capabilities between the source and the robot become increasingly critical. To address this, dynamic-aware MR methods that account for the robot's dynamics and physical limits have gained increasing attention. These methods commonly employ model-based control frameworks [2, 9, 12] as offline optimization tools to refine reference motions, ensuring dynamic feasibility with the target robot.

A second major challenge lies in developing a robust feedback controller capable of executing these motions on hardware while preserving their agility and expressiveness. In this context, motion imitation via RL is increasingly favored for its ability to produce robust control policies across diverse motion repertoires [1, 3, 6, 7, 13], while overcoming the runtime computational demands and modeling limitations of the classical model-based control approaches.

Our approach incorporates a kino-dynamic MR strategy that addresses both the kinematic and physical gap between the source and the target robot, generating a robot motion DB that consists of kinematically and dynamically feasible motion sequences using unconstrained inverse kinematics and model-based control. Meanwhile, for real-time motion tracking, we train a residual RL policy to reliably track reference motions on the physical robot.

B. Steerable and Stylistic Motion Synthesis

Generating steerable motion controllers from unstructured datasets has been extensively explored in character animation, with the goal of producing motions that are realistic, diverse, and controllable. A classical approach to this challenge is *motion matching* [14], which retrieves and blends motion segments from a given dataset based on user commands. However, motion matching inherently lacks generalization, as it essentially patchworks existing motion sequences. To overcome this limitation, recent research has increasingly turned to learning-based generative methods, including generative adversarial network (GAN)-based [15, 16] or VAE-based methods [4, 17].

GAN-based approaches have gained significant attention for generating realistic motions for physics-based characters. Adversarial Motion Priors [15] is a prominent example where an RL policy acts as the *generator* to learn natural motions by interacting with a physics simulation, guided by a *discriminator* that rewards stylistic realism. When combined with task rewards, this method enables steerable behaviors while retaining realistic motion style. This approach was further extended by Peng et al. [16], who incorporated skill embeddings to enhance scalability across a broader range of behaviors. However, these approaches often suffer from training instability and mode collapse, limiting motion diversity and requiring careful tuning.

Another line of research leverages VAEs to embed motion datasets into a structured latent space for steerable motion generation. Ling et al. [4], Won et al. [18] proposed VAE architectures where state transitions are embedded in the latent space as conditional distributions. In these frameworks, a decoder generates the next state prediction based on the previous state and a latent vector, effectively capturing motion dynamics. To enable control, an RL policy modulates these latent vectors, ensuring the resulting motion transitions align with user commands while preserving motion style.

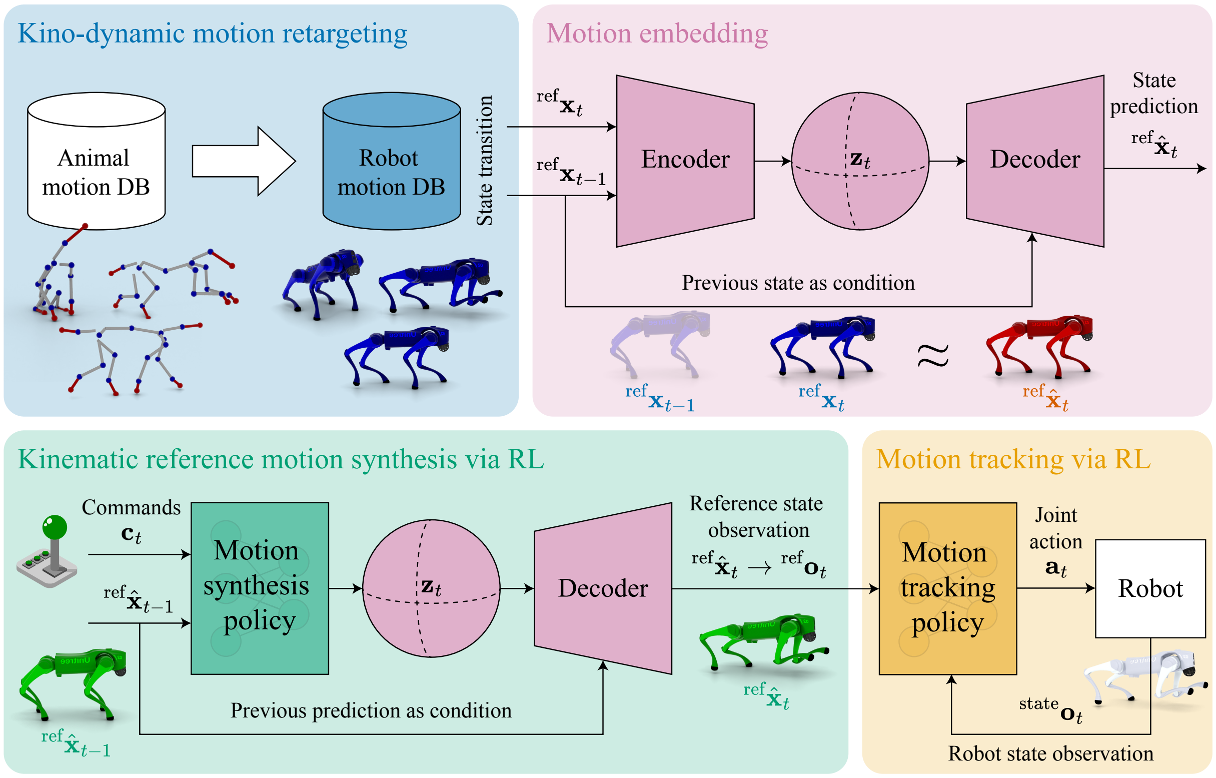
In this work, we combine a VAE and an RL policy to generate responsive and expressive motion, similar to Ling et al. [4], Won et al. [18]. To preserve the diversity of motion patterns, we use the mixture-of-experts architecture as proposed by Ling et al. [4]. Additionally, we structure the hyperspherical latent space [19], which provides a well-defined action space for the RL-based motion synthesis policy, preventing unbounded exploration during training.

III. Overview

An overview of our framework is illustrated in Figure 2. As a first step, we transform unstructured animal motion data into a robot-compatible motion DB. Specifically, we use constrained inverse kinematics and a model-based control framework to ensures the resulting motion sequences are both kinematically and dynamically feasible for the robot.

Subsequently, we embed the state transitions from the retargeted motion DB into a latent space using a *hyperspherical* VAE. The VAE is trained to reconstruct the current state ${}^{\text{ref}}\hat{\mathbf{x}}_t$, conditioned on the previous state ${}^{\text{ref}}\mathbf{x}_{t-1}$ and a latent vector $\mathbf{z}_t$. We effectively capture diverse gait patterns and smooth transitions between them in the motion DB using a mixture-of-experts (MoE) decoder architecture.

Next, a reference motion synthesis module is developed using the frozen VAE decoder. This module generates reference motions in response to user-specified velocity commands $\mathbf{c}_t$, while maintaining stylistic coherence with the motion DB. We employ an RL-based motion synthesis policy that modulates latent variables conditioned on $\mathbf{c}_t$ and the previously generated reference motion ${}^{\text{ref}}\hat{\mathbf{x}}_{t-1}$. The policy



Kino-dynamic motion retargeting
Animal motion DB
Robot motion DB
State transition
Motion embedding
$^{\text{ref}}\mathbf{x}_t$
$^{\text{ref}}\mathbf{x}_{t-1}$
Encoder
$\mathbf{z}_t$
Decoder
State prediction
$^{\text{ref}}\hat{\mathbf{x}}_t$
Previous state as condition
$^{\text{ref}}\mathbf{x}_{t-1}$
$^{\text{ref}}\mathbf{x}_t$
$\approx$
$^{\text{ref}}\hat{\mathbf{x}}_t$
Kinematic reference motion synthesis via RL
Commands
$\mathbf{c}_t$
$^{\text{ref}}\hat{\mathbf{x}}_{t-1}$
Motion synthesis policy
$\mathbf{z}_t$
Decoder
Reference state observation
$^{\text{ref}}\hat{\mathbf{x}}_t \rightarrow {}^{\text{ref}}\mathbf{o}_t$
Previous prediction as condition
$^{\text{ref}}\hat{\mathbf{x}}_{t-1}$
$^{\text{ref}}\hat{\mathbf{x}}_t$
Motion tracking via RL
Motion tracking policy
Joint action
$\mathbf{a}_t$
Robot
$^{\text{state}}\mathbf{o}_t$
Robot state observation

Fig. 2: Overview of the framework. An animal motion DB is first transformed into a robot motion DB using kino-dynamic motion retargeting (in **blue**). Next, each state transition in the motion DB is embedded into a latent space using a VAE (in **purple**). The trained decoder, combined with an RL-based motion synthesis policy produces a new reference motion in response to velocity commands (in **green**). Finally, the reference motion is tracked by an RL controller (in **orange**).

is trained to navigate the latent space, producing new reference motion $^{\text{ref}}\hat{\mathbf{x}}_t$ that effectively follows arbitrarily sampled velocity commands during training.

Finally, the synthesized reference motion $^{\text{ref}}\hat{\mathbf{x}}_t$ is provided as a tracking target to an RL feedback tracking controller, trained to robustly track the reference motion on a physical robot. At runtime, we deploy the pipeline comprising the motion synthesis policy, the VAE decoder, and the RL tracking controller to enable the robot to interactively respond to user commands. The subsequent chapters describe each stage of this framework in detail.

IV. Kino-dynamic Motion Retargeting

We use a *kino-dynamic* MR approach that combines constrained inverse kinematics (IK) with a model-based control framework. This ensures that motions retargeted from an animal motion DB are both kinematically and dynamically feasible for the robot. Similar to the previous work by Yoon et al. [2], we decouple the MR process into two stages: a kinematics stage and a dynamics stage.

In the kinematics stage, we process a source animal's motion pose-by-pose starting from the first time instance. At each time step, we extract the source base position $[^{\text{src}}x, {}^{\text{src}}y, {}^{\text{src}}z]$, base roll, pitch and yaw angles $[^{\text{src}}\phi, {}^{\text{src}}\theta, {}^{\text{src}}\psi]$, and limb vectors $^{\text{src}}_{\mathcal{B}}\mathbf{e}_i$ with $i \in \{1,2,3,4\}$. The limb vectors are unit vectors, expressed in the base frame $\{\mathcal{B}\}$, pointing from the shoulders to the corresponding foot. Additionally, we compute the ground-projected forward velocity, $^{\text{src}}v_{\text{fwd}}$, sideway velocities $^{\text{src}}v_{\text{side}}$, and yaw rate $^{\text{src}}\dot{\psi}$ of the source; these quantities are used as 2D velocity components for subsequent processing.

We adjust the base height, roll, and pitch components by applying scaling factors $\alpha_{(\cdot)} \in \mathbb{R}$, and transfer to the robot:

$$\begin{bmatrix} ^{\text{tgt}}z & ^{\text{tgt}}\phi & ^{\text{tgt}}\theta \end{bmatrix} = \begin{bmatrix} \alpha_z \, ^{\text{src}}z & \alpha_\phi \, ^{\text{src}}\phi & \alpha_\theta \, ^{\text{src}}\theta \end{bmatrix}. \quad (1)$$

Additionally, we scale and numerically integrate the 2D velocity components for the remaining base pose components:

$$\begin{bmatrix} ^{\text{tgt}}x \\ ^{\text{tgt}}y \\ ^{\text{tgt}}\psi \end{bmatrix} = \begin{bmatrix} ^{\text{tgt}}x^- \\ ^{\text{tgt}}y^- \\ ^{\text{tgt}}\psi^- \end{bmatrix} + \Delta t \mathbf{R}_z(^{\text{tgt}}\psi^-) \begin{bmatrix} \alpha_{\text{fwd}} \, ^{\text{tgt}}v^-_{\text{fwd}} \\ \alpha_{\text{side}} \, ^{\text{tgt}}v^-_{\text{side}} \\ \alpha_{\dot{\psi}} \, ^{\text{tgt}}\dot{\psi}^- \end{bmatrix}, \quad (2)$$

where the superscript $(\cdot)^-$ denotes the value at the previous timestep, and $\mathbf{R}_z(\cdot) \in \mathbb{R}^{3\times3}$ is the rotation matrix about z-axis. Subsequently, we compute the robot's foot positions by scaling limb vectors with the factor $\boldsymbol{\alpha}_{\text{limb}} \in \mathbb{R}^3$ and adding them to the shoulder positions:

$$^{\text{tgt}}\mathbf{r}_{\text{foot},i} = {}^{\text{tgt}}\mathbf{r}_{\text{shoulder},i} + {}^{\text{tgt}}\mathbf{R}_{\mathcal{WB}}(\boldsymbol{\alpha}_{\text{limb}} \odot {}^{\text{src}}_{\mathcal{B}}\mathbf{e}_i), \quad (3)$$

where $\odot$ denotes element-wise multiplication and $^{\text{tgt}}\mathbf{R}_{\mathcal{WB}}$ is the rotational matrix of the robot base. The values of the scaling factors used in our experiments are listed in Table I.

After constructing base pose and foot positions, we can use a standard IK solver to compute generalized coordinates

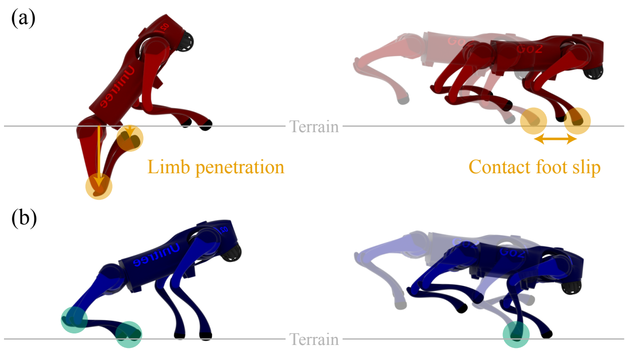


Fig. 3: (**a**) Limb penetration and contact foot slips introduced by UVM. (**b**) Our kino-dynamic MR removes these artifacts, and ensure both kinematic and dynamic feasibility.

of the robot ${}^{\text{tgt}}\mathbf{q}$ which allow us to build full pose of the robot [10]. However, this scale-and-transfer method—often refer to as unit vector method (UVM)—introduces kinematic artifacts such as contact foot slips, limb penetrations, and violation of joint limits as illustrated in Figure 3.

To address these issues, we formulate a *constrained* IK problem that enforces limb and joint constraints as follows:

$$\min_{\mathbf{q}} \; \|{}^{\text{tgt}}\mathbf{q}_{\text{base}} \ominus \mathbf{q}_{\text{base}}\|^2 + \sum_{k\in\text{swing}} \|{}^{\text{tgt}}\mathbf{r}_{\text{foot},k} - \text{FK}_{\text{foot},k}(\mathbf{q})\|^2$$

$$\begin{aligned} \text{s.t.} \quad & \text{FK}_{\text{foot},j}(\mathbf{q}) = \mathbf{r}_{\text{anc},j} && \forall j \in \text{stance} && \text{(4a)} \\ & \text{FK}_{\text{foot},k}(\mathbf{q})_z > 0 && \forall k \in \text{swing} && \text{(4b)} \\ & \text{FK}_{\text{knee},i}(\mathbf{q})_z > 0 && \forall i \in \{1,2,3,4\} && \text{(4c)} \\ & \bar{\mathbf{q}} > \mathbf{q} > \underline{\mathbf{q}}. && && \text{(4d)} \end{aligned}$$

Here, $\text{FK}_{\text{foot},i}(\cdot)$ and $\text{FK}_{\text{knee},i}(\cdot)$ are forward kinematics functions mapping the robot's generalized coordinates to the world-frame position of the i-th foot and knee, respectively. The operator $\ominus$ denotes the substraction between base pose compoenents of generalized coordinates, performing direct subtraction for positions and computing a scalar error from the quaternion difference for orientation.

The objective function of Equation (4) consists of two terms: 1) the error between the base pose components of generalized coordinates variable ${}^{\text{tgt}}\mathbf{q}_{\text{base}}$ and its IK target $\mathbf{q}_{\text{base}}$, and 2) the positional error between the swing feet and their respective IK targets. Importantly, the constraint Equation (4a) fixes the position of each stance foot j to an anchor position $\mathbf{r}_{\text{anc},j}$, where the z-component is set to the terrain height, and $x-$ and $y-$components are taken from the foot's position when the current contact was initially detected. This constraint prevents footslip and foot penetration in the retargeted motion. Additionally, Equation (4b) ensures that the swing foot is always above the terrain. Simiarly, Equation (4c) ensures that every knee is always above the terrain. Finally, Equation (4d) ensures that the generalized coordinates respect their limits, such as the joint limits.

The optimal solution from this optimization is used as a kinematically retargeted motion ${}^{\text{kin}}\mathbf{q}$, and processing the entire source motion frame by frame yields the full retargeted motion sequence ${}^{\text{kin}}\mathbf{q}_{1:N}$ where N is the sequence length.

TABLE I: Default scaling factors used in the MR kinematic stage in our experiments.

α_z	α_ϕ	α_θ	α_{fwd}	α_{side}	$\alpha_{\dot\psi}$	$\boldsymbol{\alpha}_{\text{limb}}$
0.81	1.0	1.0	0.6	0.6	1.0	$[0.6, 0.7, 0.81]$

After obtaining ${}^{\text{kin}}\mathbf{q}_{1:N}$, we proceed to the dynamics stage to ensure dynamic feasibility using a model-based control framework—specifically, model predictive control (MPC). At a high-level, this process is formulated as a trajectory tracking problem, where the objective is to follow ${}^{\text{kin}}\mathbf{q}_{1:N}$ by minimizing the generalized coordinate error over a time horizon T in a receding horizon manner. The control input is the robot's joint torque, bound by the robot's actuation limits, and the system state $\mathbf{x}$ contains the generalized coordinates and velocities: $\mathbf{x} := [\mathbf{q}, \dot{\mathbf{q}}]$.

We implement this MPC using the *MJPC* framework [20], with the *iLQG* solver [21], which leverages the full-body robot model and contact model of the *MuJoCo* simulator [22] to numerically compute the derivative information required by the solver. In our experiment, we set the time horizon to $T = 2.0\,\text{s}$ with a discretization time step of $0.01\,\text{s}$.

The resulting MPC trajectory serves as our kino-dynamically retargeted motion ${}^{\text{ref}}\mathbf{x}_{0:N}$ and is stored in our retargeted motion DB. Although solving this MPC problem is computationally expensive, MR is performed offline to generate a robot motion database for downstream learning tasks, which does not require real-time execution.

V. Motion Synthesis

The motion synthesis module is a central component of our framework, responsible for preserving the stylistic quality of motions in the DB, while capturing and reproducing their diversity in response to user-specified velocity commands. To achieve this, we adopt a VAE-based motion synthesis approach inspired by Ling et al. [4], which effectively models multiple motion modes by leveraging a MoE architecture. This approach comprises two substages: motion embedding and training an RL-based motion synthesis policy.

A. Motion Embedding

Firstly, we embed state transitions present in the motion DB into a structured latent space, by training a VAE to reconstruct the latter state in each transition pair. The input and output of the VAE are defined using local components of the robot state. Specifically, we introduce a ground-projected base frame $\{\mathcal{P}\}$, and define the VAE state vector as

$${}^{\text{vae}}\mathbf{x} := \begin{bmatrix} {}_{\mathcal{P}}z & {}_{\mathcal{P}}\mathbf{h} & {}_{\mathcal{P}}\mathbf{v} & {}_{\mathcal{P}}\mathbf{w} & {}_{\mathcal{P}}\mathbf{r}_{\text{feet}} & \boldsymbol{\theta} & \dot{\boldsymbol{\theta}} \end{bmatrix} \in \mathbb{R}^{49}, \quad (5)$$

which concatenates the base height, base orientation, base linear and angular velocities, foot positions relative to $\mathcal{P}$, joint positions, and joint velocities. The base orientation ${}_{\mathcal{P}}\mathbf{h} \in \mathbb{R}^6$ is represented using the $x-$ and $z-$axis vectors of the base frame, expressed in $\{\mathcal{P}\}$ [23].

The VAE encoder takes as input a pair of consecutive states $({}^{\text{vae}}\mathbf{x}_{t-1}, {}^{\text{vae}}\mathbf{x}_t)$, and outputs distribution parameters of the 18-dimensional latent space. The decoder then predicts

TABLE II: PPO hyperparameters.

Number of envs	4096	Value ftn. coeff.	1.0
Batch size	24576	Entropy coeff.*	0.002
Number of epochs	5	Discount factor	0.99
Learning rate	0.0005	GAE parameter	0.95
NN Hidden layers	$[512, 256, 128]$	Clipping range	0.2
NN Activation ftn.	ELU	KL target	0.01

* We set the entropy coefficient to 0.0 for the motion synthesis policy.

the current state ${}^{\text{vae}}\hat{\mathbf{x}}_t$ conditioned on the latent vector $\mathbf{z}_t$ and the previous state ${}^{\text{vae}}\mathbf{x}_{t-1}$. We adopt a MoE architecture for the decoder, consisting of six expert networks and a gating network similar to the previous work by Ling et al. [4]. This architecture effectively captures and preserves the diversity of motion patterns in the database. The encoder and each expert network in the decoder are implemented as fully connected networks with two hidden layers of 256 units. The gaiting network is also a fully connected network, with two hidden layers of 64 units. All hidden layers use the ELU activation function.

To structure the latent space, we use a hyperspherical latent representation [19] by modeling the latent variable distribution as a von Mises-Fisher (vMF) distribution instead of a standard Gaussian distribution. This design choice is crucial for streamlining the training of the motion synthesis policy in the next stage, as it constrains the action space to the bounded surface of a hypersphere [16].

The VAE training loss is defined as:

$$\mathcal{L} = \|{}^{\text{vae}}\mathbf{x}_t - {}^{\text{vae}}\hat{\mathbf{x}}_t\|_2^2 + \beta D_{\text{KL}}\left(q\,\|\,p\right), \qquad (6)$$

where $q(\mathbf{z}_t|{}^{\text{vae}}\mathbf{x}_t, {}^{\text{vae}}\mathbf{x}_{t-1})$ is the posterior distribution approximated by the encoder and $p(\mathbf{z}_t)$ is the vMF prior. We refer the readers to the work by Davidson et al. [19] for the derivation of KL divergence term for vMF distribution. In our experiments, we set the weighting coefficient to $\beta = 0.05$.

Following Ling et al. [4], we initially train the VAE using state transitions $({}^{\text{vae}}\mathbf{x}_{t-1}, {}^{\text{vae}}\mathbf{x}_t)$ from the motion database for 20 epoch. We then gradually shift to autoregressive training over 60 epochs, where the decoder's prediction ${}^{\text{vae}}\hat{\mathbf{x}}_t$ is recursively used as the next input condition (i.e. ${}^{\text{vae}}\hat{\mathbf{x}}_t \to {}^{\text{vae}}\mathbf{x}_{t-1}$). This training strategy improves the stability of sequence prediction in downstream tasks.

B. Motion Synthesis Policy

In the second stage, we train the motion synthesis policy using RL to navigate the hyperspherical latent space constructed in the previous stage and generate states that follow the user's velocity commands.

The policy observes the user's forward and turning speed commands $\mathbf{c}_t := [c_{\text{fwd},t}, c_{\text{turn},t}]$ and the previously generated motion state ${}^{\text{vae}}\hat{\mathbf{x}}_{t-1}$. The action is defined as a vector $\tilde{\mathbf{z}}_t \in \mathbb{R}^{18}$ which is later projected onto the hyperspherical latent space by $\mathbf{z}_t = \tilde{\mathbf{z}}_t / \|\tilde{\mathbf{z}}_t\|_2$.

The policy is trained with Proximal Policy Optimization (PPO) algorithm [24] using a reward function designed to align the ground-projected forward speed of the robot's base v_{fwd} and yaw rate $\dot{\psi}$ with their commanded values:

$$r = \exp\left(-\frac{(v_{\text{fwd}} - c_{\text{fwd}})^2}{0.25} - \frac{(\dot{\psi} - c_{\text{turn}})^2}{0.1}\right) \qquad (7)$$

The trained RL policy, together with the decoder, constitutes the motion synthesis module, which generates reference motions in real time in response to user commands. The list of the PPO hyperparameters used is provided in Table II.

TABLE III: Motion tracking policy reward hyperparameters.

Reward terms r_{y}	Sensitivity σ_{y}
Base linear velocity r_{v}	0.2
Base angular velocity r_{w}	0.25
Base height r_z	0.1
Base orientation $r_{\phi\theta}$	0.8
Feet position r_{feet}	$[0.3, 0.3, 0.1]^*$
Global position r_{xy}	0.5
Global orientation r_{h}	0.5
Action rate $r_{\Delta\text{a}}$	2.0
Action scale r_{a}	10.0
Feet slip r_{slip}	0.1

* Non-scalar values are applied element-wise.

VI. Motion Tracking via Residual Policy

As a final step, we use RL to train a policy that tracks a synthesized reference motion robustly. We design the RL policy to generate a residual joint position action added to the joint position components of the reference motion. Additionally, to facilitate the training for highly dynamic gait patterns such as *Gallop*, we adopt the asymmetric actor critic approach [25] which incorporates privileged information in the observation of the critic, to better guide the training.

A. Observation and Action Space

At timestep t, the policy (actor) takes as input a noisy robot's state observation denoted as ${}^{\text{state}}\mathbf{o}_t \in \mathbb{R}^{42}$, which includes the base orientation (represented as a gravity vector in the base frame), angular velocity, joint angles, and previous joint commands of the robot, along with the reference base height, orientation, linear velocity, angular velocity, joint positions, joint velocities and foot heights denoted as ${}^{\text{ref}}\mathbf{o}_t \in \mathbb{R}^{26}$. The base velocity components are expressed in the base frame. The full observation vector is formed as a stack of the state and reference motion observation with their history, with the history length $H = 4$. Additionally, the latent vector $\mathbf{z}_t \in \mathbb{R}^{18}$ from the motion synthesis module is included:

$$\mathbf{o}_t = \begin{bmatrix} {}^{\text{state}}\mathbf{o}_{t-H:t} & {}^{\text{ref}}\mathbf{o}_{t-H:t} & \mathbf{z}_t \end{bmatrix} \in \mathbb{R}^{358}. \qquad (8)$$

As animal motions involve frequent flying phases and irregular stepping patterns, estimating the robot's base height and linear velocity becomes challenging. Therefore, we chose to exclude these components from the observation.

The critic takes as input the noise-free version of the policy observations, along with privileged information including the base height, linear velocity, actual and reference foot positions relative to the base, and foot velocities. Additionally, it receives the position and orientation error of the base in

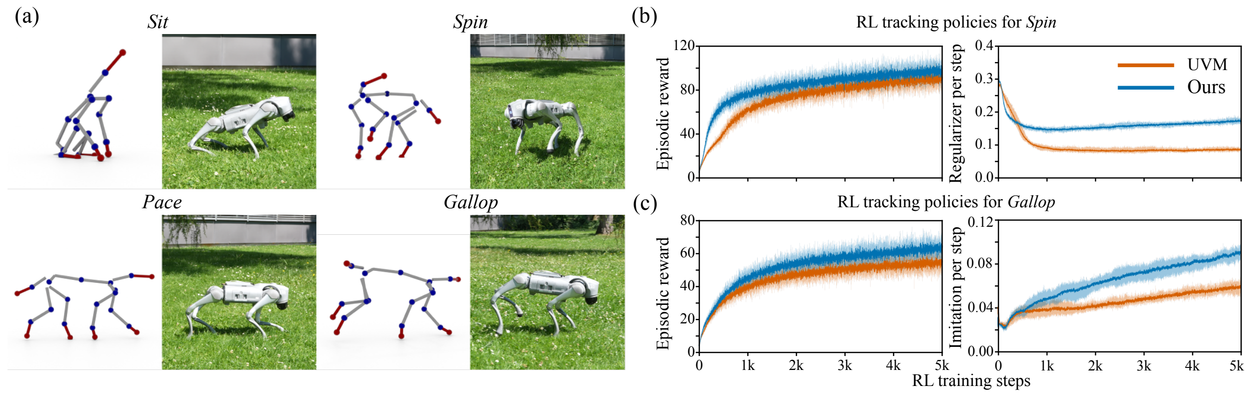


Fig. 4: (**a**) Our kino-dynamic MR enables reliable transfer of dog motion sequences to the real-world robot. We evaluate its effectiveness by comparing RL training curves against the UVM baseline method for two motions: *Spin* (**b**) and *Gallop* (**c**).

the world frame to encourage exploration for highly dynamic motions.

The output of the policy is residual joint actions $\mathbf{a} \in \mathbb{R}^{12}$, which is added to the reference joint angles ${}^{\text{ref}}\boldsymbol{\theta}_t$ with a scaling factor and set as the PD targets of the robot joints, ${}^{\text{PD}}\boldsymbol{\theta}_t = {}^{\text{ref}}\boldsymbol{\theta}_t + 0.15 \cdot \mathbf{a}_t$.

B. Reward Design

The reward function is formulated as a weighted sum of the imitation reward r_I, world frame pose reward $r_{\mathcal{W}}$, and regularizer r_R, with corresponding weights w_I, $w_{\mathcal{W}}$, and w_R:

$$r = w_I \cdot r_I + w_{\mathcal{W}} \cdot r_{\mathcal{W}} + w_R \cdot r_R. \tag{9}$$

The imitation reward encourages the robot to follow the reference motion, and is defined as:

$$r_I = r_z \cdot r_{\text{v}} \cdot r_{\text{w}} \cdot r_{\phi\theta} \cdot r_{\text{feet}} \tag{10}$$

where r_z matches the base height, r_{v} matches xy components of base linear velocity, r_{w} matches z component of base angular velocities, $r_{\phi\theta}$ matches the roll and pitch angles of the base, and finally r_{feet} matches the relative foot positions with respect to base. All components are expressed in the robot's base frame.

The world frame pose reward matches the (x, y) position and orientation of the robot's base and that of the reference, expressed in the world frame: $r_{\mathcal{W}} = r_{xy} \cdot r_{\text{h}}$. This reward term improves base motion tracking for dynamic movements at high velocities.

Finally, the regularizer term $r_R = r_{\Delta\text{a}} \cdot r_{\text{a}} \cdot r_{\text{slip}}$ consists of penalties on action rate, action value, and foot slip.

Note that the subterms of r_I, $r_{\mathcal{W}}$, and r_R are multiplied together, where each subterm follows the following form:

$$r_{\text{y}} = \exp\left(-\left\|\frac{\hat{\mathbf{y}} - \mathbf{y}}{\sigma_{\text{y}}}\right\|^2\right), \tag{11}$$

where $\hat{\mathbf{y}}$ is the desired value of the robot quantity $\mathbf{y}$, and σ_{y} denotes sensitivity parameter. The detailed reward hyperparameters are listed in Table III.

C. Other Training Details

The tracking policy is trained in a physically simulated environment using *NVIDIA IsaacLab* [26]. The simulated *Unitree Go2* robot is equipped with a joint PD controller with $K_p = 30$ and $K_d = 0.5$. The policy is queried at $50\,\text{Hz}$, with each control action executed over four simulation steps.

To facilitate exploration around the reference motion, we implemented reference state initialization [27]. Additionally, we apply domain randomization to improve the general robustness of the tracking policies against the sim-to-real gap and external disturbances. Specifically, we randomize the friction coefficient within the range $[0.5, 1.5]$, vary the mass of each link by $\pm 10\%$, perturb the center of mass of each link by up to $0.05\,\text{m}$, and apply random external perturbations to the robot base every 1.5 to $2.5\,\text{s}$.

The tracking policy is also trained using PPO with the same set of hyperparameters as the motion synthesis policy, except for the entropy coefficient, which is set to 0.002.

VII. Results

We conducted a series of simulation and real-world experiments to evaluate each component of our framework and to demonstrate the full control pipeline developed using the proposed approach. In our experiments, we used a subset of a dog motion dataset [28], consisting of 13076 samples along with their left-right mirrored counterparts.

In the first set of experiments, we evaluate our kino-dynamic MR strategy for reliable motion imitation. We selected three representative animal motion sequences—*Sit-pace-sit*, *Spin*, and *Gallop*—and retargeted them using both our method and the UVM baseline. Motion tracking policies were then trained to execute these sequences on *Go2* using five random seeds over 5000 PPO update steps.

As shown in Figure 3, the UVM baseline results in hind limb penetrations during the *Sit-pace-sit* motion, which prevents successful tracking due to persistent ground contact. Similarly, for *Spin*, the UVM-retargeted motion exhibits frequent foot slippage. Although the RL policy appears to track the motion, the per-step regularizer reward curve in Figure 4(b) reveals unstable contact behavior, making the motion less suitable for real-world deployment. For the

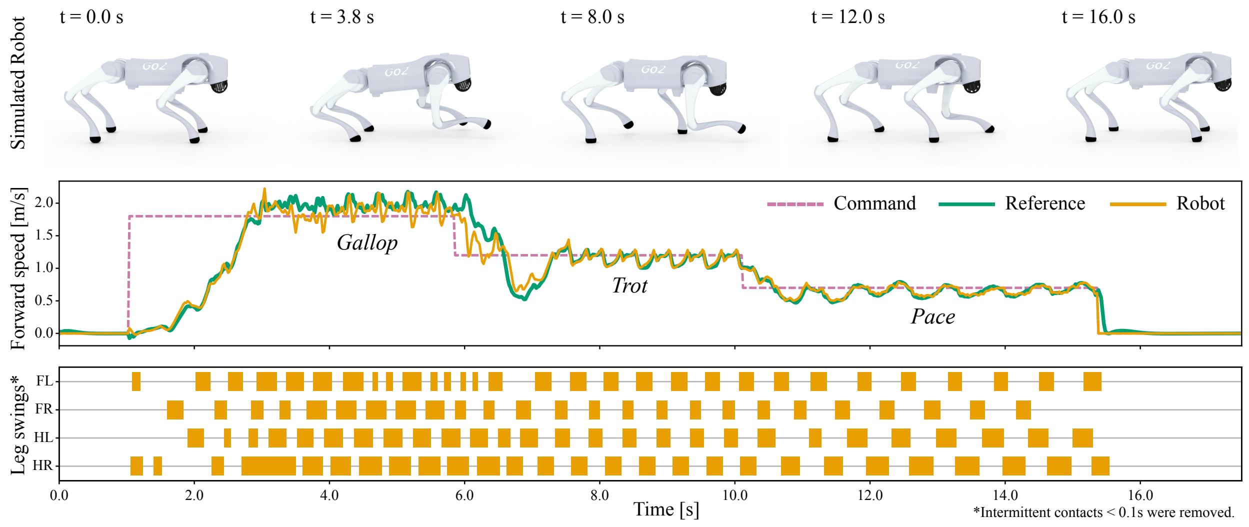


Fig. 5: Snapshots of physically simulated *Go2* (**top**) executing a motion sequence generated by our motion synthesis module in response to varying forward speed commands shown alongside the speed profile (**middle**) and leg swing timeline (**bottom**).

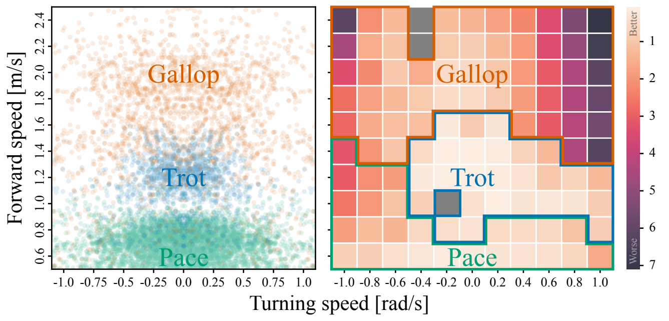


Fig. 6: Sample distribution from the motion DB visualized as a scatter plot over forward and turning speeds (**left**). Command tracking error of the motion synthesis module shown as a heatmap. Gray cells indicate the synthesized motion fails to respond to the command and remains stationary (**right**).

dynamic *Gallop* motion, the UVM-retargeted version causes the imitation reward to saturate prematurely, as shown in Figure 4(c), indicating poor tracking performance. In contrast, our kino-dynamic MR approach produces dynamically feasible motions that lead to better tracking quality and overall more reliable motion imitation.

In summary, our kino-dynamic MR effectively transforms agile and expressive animal motions into robot-compatible trajectories, enabling reliable motion execution on hardware as illustrated in Figure 4(a).

Shifting our focus to motion synthesis, we evaluate the ability of our motion synthesis module to generate motions that accurately follow velocity commands and exhibit appropriate gait transitions. Starting from a nominal standing pose, we apply a range of forward speed commands $[0.6, 2.4]\,\mathrm{m/s}$ and turning speed commands $[-1.0, 1.0]\,\mathrm{rad/s}$, and measure the tracking error over $10\,\mathrm{s}$. The heatmap in Figure 6 visualizes the weighted sum of mean squared forward speed error e_{fwd} and turning speed error e_{turn}, computed as $e_{\text{fwd}} + 10 \cdot e_{\text{turn}}$. A higher weight is applied to the turning error to balance the visualization, as its magnitude is relatively small.

The heatmap reveals that the motion synthesis module occasionally fails to respond accurately to user commands, particularly in regions where the motion data is sparse. Nonetheless, the module generally tracks the commands well and, most notably, is able to capture and reproduce the diverse gait patterns present in the dataset.

To better examine the behavior of the synthesized motion, we generated a motion sequence using our motion synthesis module under varying forward speed commands. We then trained a motion tracking policy to follow this sequence and executed it on the physically simulated *Go2*, as shown in Figure 5. As the forward speed profile indicates, the reference motion produced by the motion synthesis module responds smoothly and accurately to the commands, transitioning seamlessly from *Gallop* to *Trot* to *Pace* as the speed varies from $1.8\,\mathrm{m/s}$ to $1.2\,\mathrm{m/s}$ to $0.7\,\mathrm{m/s}$. The tracking policy successfully reproduces this motion on the robot, achieving a root mean squared base velocity error of $0.11\,\mathrm{m/s}$.

Finally, we deployed the full control pipeline—comprising the motion synthesis module and the RL tracking controller—on the robot hardware to enable responsive, real-time steering. For this experiment, the RL motion tracking policy was trained with the motion synthesis module in the loop, which generates reference motion for the policy based on randomly sampled velocity commands. Due to a stability limitations on hardware, we restricted the forward speed command range to $[0, 1.3]\,\mathrm{m/s}$, and focused our demonstrations on lower-speed locomotion.

As shown in Figure 1, the control pipeline enables the robot to navigate freely with animal-like gait patterns and demonstrates adaptive gait switching in response to velocity commands. Readers are referred to the supplementary video for comprehensive footage of the experiments [1].

[1]The video is available at `https://youtu.be/DukyUGNYf5A`

VIII. Conclusion and Future work

This work presents a learning-based framework that reproduces diverse behaviors from unstructured animal motion data. In so doing, it enables steerable control in legged robots with natural gait transitions that emerge in response to velocity commands. Our framework effectively bridges both the morphological and physical gaps between a source animal and a target robot, reliably reproducing expressive and agile animal movements while preserving the diversity of the original dataset.

During development, we identified several limitations in the methodologies adopted within our framework. Most notably, we observed that the VAE-based motion synthesis module often introduces artifacts—such as unrealistically aggressive motions—particularly at high velocity command ranges. This limitation stems from the fact that the motion synthesis module is trained purely kinematically. While prior work has explored incorporating physics into the training process [18, 17], such approaches often suffer from mode collapse, particularly for dynamic behaviors, which are inherently more difficult to learn. Addressing this issue and enabling consistent, artifact-free motion synthesis remains an important and promising direction for future research.

Additionally, we found that training a single RL-based motion tracking policy to reliably execute a wide range of motion patterns is highly challenging, as different skills often exhibit significantly different dynamic properties. For example, a policy trained to track both *Pace* and *Gallop* gaits tends to show compromised performance on *Gallop*, failing to specialize effectively. We are interested in adopting a mixture-of-policies approach, similar to the concurrent work by Chen et al. [29], which could enable more reliable execution of diverse motion behaviors on robot hardware.

A promising extension of our work is its application to humanoid robots, where natural and efficient motor skill learning through motion imitation is gaining increasing attention. In particular, we are interested in developing a loco-manipulation pipeline that supports a diverse set of skills with seamless transitions between them, adapting fluidly to both environmental context and user commands.

References

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

[2] T. Yoon, D. Kang, S. Kim, M. Ahn, S. Coros, and S. Choi, "Spatio-temporal motion retargeting for quadruped robots," 2024.

[3] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, Z. Yi, G. Qu, K. Kitani, J. Hodgins, L. J. Fan, Y. Zhu, C. Liu, and G. Shi, "Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills," 2025.

[4] H. Y. Ling, F. Zinno, G. Cheng, and M. Van De Panne, "Character controllers using motion vaes," *ACM Trans. Graph.*, vol. 39, no. 4, Aug. 2020.

[5] 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.

[6] 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," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 25–32.

[7] A. Tang, T. Hiraoka, N. Hiraoka, F. Shi, K. Kawaharazuka, K. Kojima, K. Okada, and M. Inaba, "Humanmimic: Learning natural locomotion and transitions for humanoid robot via wasserstein adversarial imitation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 13 107–13 114.

[8] 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 in quadrupedal robots using reinforcement learning and generative pre-trained models," *Nature Machine Intelligence*, vol. 6, no. 7, pp. 787–798, 2024.

[9] J. Z. Zhang, S. Yang, G. Yang, A. L. Bishop, S. Gurumurthy, D. Ramanan, and Z. Manchester, "Slomo: A general system for legged robot motion imitation from casual videos," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7154–7161, 2023.

[10] S. Choi and J. Kim, "Towards a natural motion generator: a pipeline to control a humanoid based on motion data," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4373–4380.

[11] 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.

[12] R. Grandia, F. Farshidian, E. Knoop, C. Schumacher, M. Hutter, and M. Bächer, "DOC: Differentiable Optimal Control for Retargeting Motions onto Legged Robots," *ACM Transactions On Graphics (TOG)*, vol. 42, no. 4, pp. 1–14, 2023.

[13] F. Zargarbashi, J. Cheng, D. Kang, R. Sumner, and S. Coros, "Robotkeyframing: Learning locomotion with high-level objectives via mixture of dense and sparse rewards," in *Proceedings of The 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, P. Agrawal, O. Kroemer, and W. Burgard, Eds., vol. 270. PMLR, 2025, pp. 916–932.

[14] S. Clavet, "Motion matching and the road to next-gen animation," in *Proc. of GDC*, 2016.

[15] 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.

[16] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "Ase: large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, 2022.

[17] H. Yao, Z. Song, B. Chen, and L. Liu, "Controlvae: Model-based learning of generative controllers for physics-based characters," *ACM Trans. Graph.*, vol. 41, no. 6, Nov. 2022.

[18] J. Won, D. Gopinath, and J. Hodgins, "Physics-based character controllers using conditional vaes," *ACM Trans. Graph.*, vol. 41, no. 4, Jul. 2022.

[19] T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak, "Hyperspherical variational auto-encoders," *34th Conference on Uncertainty in Artificial Intelligence (UAI-18)*, 2018.

[20] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," 2022.

[21] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4906–4913.

[22] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[23] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[25] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[26] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automa-*

tion Letters, vol. 8, no. 6, pp. 3740–3747, 2023.

[27] 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.

[28] H. Zhang, S. Starke, T. Komura, and J. Saito, “Mode-adaptive neural networks for quadruped motion control,” *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018.

[29] Z. Chen, M. Ji, X. Cheng, X. Peng, X. B. Peng, and X. Wang, “Gmt: General motion tracking for humanoid whole-body control,” 2025.