Robotics: Science and Systems 2025
Los Angeles, CA, USA, June 21–25, 2025

Discrete-Time Hybrid Automata Learning: Legged Locomotion Meets Skateboarding

Hang Liu[1] Sangli Teng[1†] Ben Liu[2] Wei Zhang[2] Maani Ghaffari[1]
[1]University of Michigan [2]Southern University of Science and Technology
†Corresponding Author: sanglit@umich.edu
Website, Code: https://umich-curly.github.io/DHAL/

Fig. 1: Demonstration of DHAL performance across various indoor and outdoor terrains, including slopes, carpets, sidewalks, step, and scenarios with additional payloads or disturbance. The controller enables the robot to perform smooth and natural skateboarding motions, with reliable mode identification and transitions under disturbances.

***Abstract*—Hybrid dynamical systems, which include continuous flow and discrete mode switching, can model robotics tasks like legged robot locomotion. Model-based methods usually depend on predefined gaits, while model-free approaches lack explicit mode-switching knowledge. Current methods identify discrete modes via segmentation before regressing continuous flow, but learning high-dimensional complex rigid body dynamics without trajectory labels or segmentation is a challenging open problem. This paper introduces Discrete-time Hybrid Automata Learning (DHAL), a framework to identify and execute mode-switching without trajectory segmentation or event function learning. Moreover, we embed it in a reinforcement learning pipeline and incorporate a beta policy distribution and a multi-critic architecture to model contact-guided motions, exemplified by a challenging quadrupedal robot skateboard task. We validate our method through sufficient real-world tests, demonstrating robust performance and mode identification consistent with human intuition in hybrid dynamical systems.**

I. INTRODUCTION

In state space representation, systems that exhibit flow-based continuous and jump-based discrete dynamics are known as hybrid dynamical systems [1]. Such systems are prevalent in many real-world environments, particularly those involving discontinuities, contact events, or mode switching—examples include legged robotics, power systems (e.g., thermostat systems [2], DC-DC converters), and even neurobiological models such as the integrate-and-fire neuron [3].

In robotics, the walking behavior of bipedal robots is a quintessential example of a hybrid dynamical system. In model-based control, the hybrid automata framework has been proposed as a powerful tool to describe systems encompassing discrete and continuous dynamics [4, 5]. This framework has been widely adopted for behavior planning [6] and legged locomotion. However, these approaches typically rely on simplified dynamic models, and contact events are often predefined, as in manually crafted walking gaits.

With the rise of data-driven paradigms and world models, recent research has begun exploring learning-based approaches for hybrid dynamics [7, 8, 9]. However, these efforts focus on low-dimensional or single-trajectory scenarios, and inefficient

Fig. 2: The potholes on the downhill slope caused the robot dog's right front leg to get stuck, preventing it from smoothly getting onto the board. Both of its hind legs even lost contact with the board. Nevertheless, the policy could still guide the robot dog to jump back onto the board and complete the recovery behavior.

training. Designing methods that can generalize to high-dimensional systems, handle complex real-world dynamics, and effectively identify discrete modes remains an open problem.

Towards this goal, we propose a generalized approach for mode identification and prediction in hybrid dynamical systems. We design a set of hybrid dynamics modules that integrate discrete hybrid automata(DHA) with a variational autoencoder (VAE) framework, enabling the system to learn mode identification and flow dynamics heuristically. To demonstrate its effectiveness, we evaluated our approach in contact-guided scenarios involving complex sequences of contact events [10].

Designing such contact-rich scenarios is highly non-trivial. Unlike model-based methods, model-free reinforcement learning (RL) has shown promise in solving optimal control problems (OCPs) by modeling dynamics as a Markov Decision Process. Model-free RL requires minimal assumptions and can be applied to various tasks across diverse dynamical systems [11, 12]. We integrate a multi-critic architecture with a Beta distribution policy to address contact-rich tasks and embed our hybrid dynamics system into the RL pipeline.

We tackle the challenging task of enabling a quadrupedal robot to skateboard. This task exemplifies a highly dynamic, contact-guided, and hybrid underactuated system, requiring precise handling of mode transitions and contact events. The key contributions of this work are summarized as follows:

1) **Discrete hybrid automata framework**: We propose a discrete hybrid automata framework that eliminates the need for explicit trajectory segmentation or event labeling in mode identification and dynamics learning.
2) **Contact-guided task design**: We combined the multi-critic architecture and the beta distribution to effectively address the contact-guided problem in hybrid systems.
3) **Sim2Real of underactuated skateboarding motion**: We achieved agile and robust sim-to-real performance in the highly underactuated and hybrid task of skateboard motion.

II. RELATED WORK

A. Legged Robot Control

The Model Predictive Control (MPC) [13, 14, 15] with simplified single rigid body has been successfully applied to motion planning of legged robot [16, 17, 18, 19], achieving robust locomotion on flat ground under diverse gait patterns. Corbères et al. [16], Grandia et al. [17] further integrated motion planning and perception, enabling quadruped robots to navigate complex terrains. However, such approaches depend highly on accurate global state estimation, which poses limitations in outdoor and long-range scenarios. Additionally, the gait pattern of the model-based approach is designed manually, which can not scale in complicated scenarios.

In contrast to the model-based approach, model-free reinforcement learning (RL) has demonstrated remarkable capabilities in legged robot control, including high-speed locomotion [20, 21, 22, 23], complex terrain traversal [24, 25, 26, 27], manipulation, and interaction [28, 29, 30]. The primary focus of RL-based quadruped control in recent years is on the paradigm of sim-to-real transfer [31, 26], safe reinforcement learning [22], and gait control [32, 33, 34]. In this paper, we primarily focus on exploring sparse motion patterns and embedding hybrid dynamics learning.

B. Contact-guided locomotion pattern

The hybrid nature of contact dynamics makes it challenging to synthesize optimal motions for contact-rich tasks [35, 36, 37]. In model-based methods, the discontinuities introduced by contact create obstacles for gradient-based optimization. Kim et al. [38] and [39] address this issue by formulating it as a linear complementarity problem (LCP) and relaxing the complementarity constraints. In contrast, Westervelt et al. [4] adopts a hybrid dynamic system combined with automata to handle contact.

In the realm of reinforcement learning (RL), the specification of sparse or contact-guided motion patterns has not been widely investigated or solved, such as explicitly prescribing contact-based motion [10] or relying solely on key frame trajectory tracking [40]. Considering real-world robot collisions and constraints, sparse reward design in a high-dimensional sampling space severely limits effective exploration [10]. In contrast, under a multi-critic framework [40] combined with a Beta action distribution [41], this paper achieves contact-specified skateboarding motion on a quadruped robot, with on-the-fly adjustments of the gait pattern.

C. Hybrid Dynamics

Hybrid dynamic systems are employed to describe systems that feature continuous states and discrete modes, and they are widely used in model-based control and cyber-physical systems [42]. For instance, floating-base robots are often treated as hybrid dynamic systems due to the discontinuities and

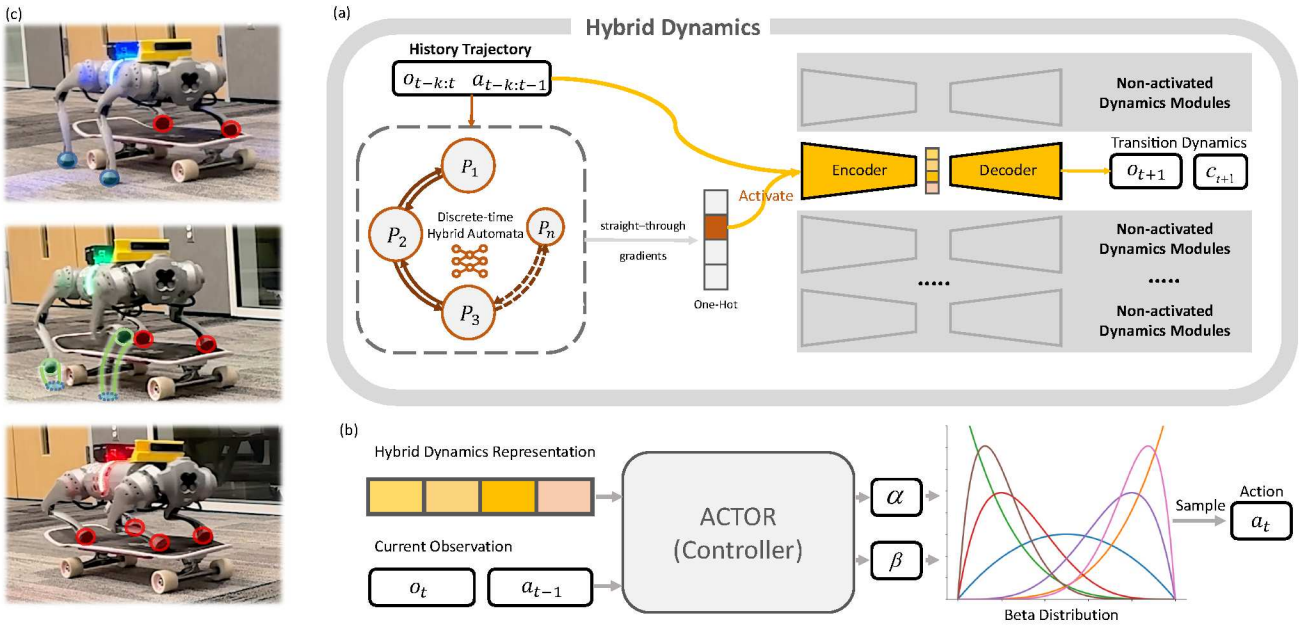

Fig. 3: Discrete-time Hybrid Dynamics Learning (DHAL) Framework: (a) During training, the network learns to select the mode and activate the corresponding dynamics module (yellow-highlighted) to predict transition dynamics and contact. Here, P_i represents the probability of the robot being in mode i at time t. (b) The temporal features extracted by the encoder are combined with the current state and last action into the actor. The actor updates α, β, which define the probability density function of the Beta distribution, and then samples joint actions from the Beta distribution. (c) In a real-world deployment, we use different LED colors to indicate the active modes, showcasing smooth transitions and mode-specific behaviors.

jumps caused by contact. Recently, Ly and Lipson [7], Chen et al. [8], Poli et al. [9], Teng et al. [43] have leveraged data-driven approaches to construct either discrete or continuous hybrid dynamic systems.

Unlike previous work, we integrate reinforcement learning with hybrid dynamics, enabling the robot to learn a hybrid dynamics automata for explicit mode switching and control without requiring labels or segmentation.

III. BACKGROUND AND PROBLEM SETTING

For clarity and reference, Table I provides an overview of the key symbols and abbreviations used throughout this paper.

A. Markov Decision Process

We model the robot control problem as an infinite-horizon partially observable Markov decision process (POMDP), composed by tuple $\mathcal{M} = (S, \mathcal{O}, A, p, r, \gamma)$, with $s_t \in \mathcal{S}$ the full states, $o_t \in \mathcal{O}$ the partial observation of the agents from the environment, $a_t \in \mathcal{A}$ the action the agent can take to interact with the environments, and $p(s_{t+1}|s_t, a_t)$ the transition function of state s_t. Each transition is rewarded by a reward function $r : S \times A \to \mathbb{R}$ with γ representing a discount factor. The optimization objective of reinforcement learning is to maximize the expected total return $\mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$.

B. Discrete Hybrid Dynamical System

The hybrid dynamical system involves discrete and continuous dynamics or states. The system has a continuous flow in each discrete mode and can jump between these modes [44, 9]. Although using a set of ordinary differential equations (ODEs) with transition maps modeled by neural networks has shown some promising results [9, 8], the continuous integration is computationally expensive and challenging to apply in real-world robot control, which is a digital control system. In this work, inspired by Borrelli et al. [45] and Ly and Lipson [7], we adopt *discrete hybrid automata* to model the dynamics of legged robots, which naturally exhibit hybrid behaviors.

For embedding discrete hybrid automata, we utilize the concept of switched systems to model the hybrid dynamics for legged robots. The dynamics of the legged robot in each mode can be described as

$$s_{t+1} = f^{i_t}(s_t, a_t), \tag{1}$$

where $s_t \in \mathbb{R}^n$ is the states, $a_t \in \mathbb{R}^m$ is the input, $i_t \in \mathcal{I} = \{1, 2, ..., K\}$ is the modes at time step t, $f^{i_t} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the dynamics on mode i. The mode i_t is determined by an extra mode selector. We assume that all system states are continuous-valued, not considering any discrete-valued state. Dynamics for each mode is unique, i.e., $f^i \neq f^j, \forall i, j \in \mathcal{I}$. The maximum possible number of mode K is assumed to be known.

In this work, we use neural networks to model both the dynamics and the automata (mode selector), aiming to extract a latent representation that informs the actor. To maintain consistency with the stochastic nature of the MDP, we utilize a β-VAE to model the state transition in equation (1). Unlike hybrid systems described by continuous flow [44, 9], we omit the explicit jump mapping between different modes, as it is

captured within the discrete-time dynamics (1).

C. Environment Design

To demonstrate the effectiveness of our approach, we aimed to select a challenging environment that involves complex mode transitions and contact-rich dynamics. Inspired by the real-world example of dogs learning to ride skateboards, we identified the task of a robotic dog skateboarding as a highly demanding scenario. This task presents distinct hybrid dynamics challenges, such as the significant differences between the gliding and pushing modes. We believe this represents a worthwhile and meaningful challenge for validating our method. Our method is not limited to the skateboarding task but can also be expanded to other scenarios.

In this paper, we conducted experiments using the Unitree Go1 robot. The skateboard used in our experiments measures 800 mm × 254 mm × 110 mm. The Unitree Go1 is equipped with 12 actuated joints. Although it lacks the spinal degrees of freedom in real quadrupedal animals, our experimental results demonstrate that it can still effectively control skateboard movement.

To simplify the design of the system, we connected the end of the Unitree Go1 left forelimb to the skateboard using a spherical joint in simulation, providing passive degrees of freedom along the x, y, and z axes. Unlike Chen et al. [46], the wheels of the skateboard can only passively rotate around their respective axes. Additionally, we simulated the truck mechanism of the skateboard in the simulation using a position PD controller to replicate its mechanical behavior.

TABLE I: Important symbols and abbreviations

Meaning	Symbol
POMDP	
Full State	s_t
Partial Observation	o_t
Action	a_t
State Space	$\mathcal{S}$
Action Space	$\mathcal{A}$
Discount Factor	γ
Hybrid Dynamics System	
Discrete-time dynamics	f^{i_t}
Mode index	i_t
Number of modes	K
Jacobian	J
Probability of each mode	p
mode indicator vector	δ
maximum number of modes	$K = \lvert\delta\rvert$
Environment	
Joint Position	$q = \begin{bmatrix} q_{Hip,Thigh,Calf}^{FL/FR/RL/RR} \end{bmatrix}$
Joint Velocity	$\dot{q} = \begin{bmatrix} \dot{q}_{Hip,Thigh,Calf}^{FL/FR/RL/RR} \end{bmatrix}$
Go1 Base Roll, Pitch,Yaw	ϕ, θ, ψ
Go1 Base angular velocity	$\omega_x, \omega_y, \omega_z$
Gravity	$g_{x,y,z}$
Action	$a = \begin{bmatrix} a_{Hip,Thigh,Calf}^{FL/FR/RL/RR} \end{bmatrix}$
Phase	Φ
Command	$cmd = [c_x, c_{yaw}]$
Proprioception	$o = [q, \dot{q}, g_{x,y,z}, \phi, \theta, \psi, \omega_x, \omega_y, \omega_z]$
Contact	c
Torque	τ

IV. METHODS

We introduce DHAL, as shown in Fig. 3, to illustrate the proposed controller [4] that leverages the model-based hybrid dynamics. More specifically:

i. **Discrete-time Hybrid Automata**, a discrete mode selector, to identify the one-hot latent mode of the system at each time step.

ii. **Dynamics Encoder**, based on the mode z chosen from discrete-time hybrid automata, the chosen dynamics encoder will be activated and get a tight representation of flow dynamics.

iii. **Dynamics Decoder**, to decode the representation and predict transition dynamics o_{t+1} and contact event c_{t+1}.

iv. **Controller**, based on the tight representation from the dynamics encoder and observation at the current moment, controls the robot to perform skateboarding.

A. Discrete Neural Hybrid Automata

Previous research has explored modifications to the locomotion and manipulation framework, such as incorporating estimators to predict transition dynamics [27, 47, 48, 49]. The dynamics of legged robots inherently exhibit hybrid behavior due to contact events, as illustrated in Fig. 4. We model the contact as a perfect inelastic collision, which means the velocity of the contact point instantaneously drops to zero from a nonzero value. Consequently, the state also undergoes a discrete jump, resulting in a hybrid dynamics formulation. To illustrate this, consider a simplified example of a single 3-DoF leg attached to a fixed base. The velocity of the contact point is given by

$$v_c = J_c(q)\dot{q}, \tag{2}$$

where $v_c \in \mathbb{R}^3$ is the linear velocity of the contact point, $J_c(q) \in \mathbb{R}^{3\times3}$ is the Jacobian, $\dot{q} \in \mathbb{R}^3$ is the joint angular velocity. In most cases, $J_c(q)$ is invertible, implying that a discontinuity in v_c directly induces a discontinuity in $\dot{q}$. This simple example highlights the inherent hybrid nature of legged robot dynamics.

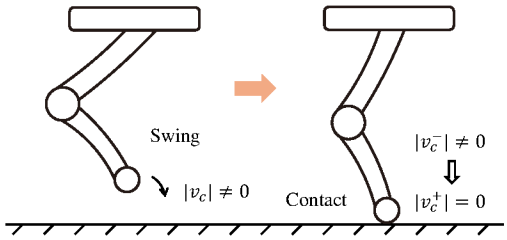

Fig. 4: Switching of a hybrid system with inelastic collision. When the leg contacts the ground, the linear velocity will abruptly drop to zero.

In this work, we aim to extract a latent representation of the dynamics for the controller. To achieve this, we model the dynamics for each mode and design a discrete-time hybrid automata (DHA) to determine which dynamics is active at any

given time, as illustrated in Fig. 3. Since only partial observations o_t are available in practice, we consider partial discrete-time dynamics rather than the full state s_t. As discussed in Sec. III-B, we employ a β-VAE instead of a deterministic dynamics model to better align with the stochastic nature of reinforcement learning. This process can be expressed by

$$\begin{aligned} i_t &= f_{\texttt{DHA}}(o_{t-k:t}, a_{t-k-1:t-1}; \theta_{\texttt{DHA}}), \\ o_{t+1}, c_{t+1} &= f_{\text{dec}}^{i_t} \circ f_{\text{enc}}^{i_t}(o_{t-k:t}, a_{t-k-1:t-1}; \theta_{\texttt{vae}}), \end{aligned} \tag{3}$$

where $i_t \in \{1, 2, ..., K\}$ is the mode for current time t, which is determined by a state-action sequence $(o_{t-k:t}, a_{t-k-1:t-1})$. $c_{t+1} \in \mathcal{S}$ represents the contact information for each foot, see (7). The maximal number of modes is pre-defined. The next partial state o_{t+1} and contact information c_{t+1} are determined by a β-VAE at mode i, taking the same input as the hybrid automata. A key advantage of the β-VAE is that it directly provides a latent representation of the dynamics. The hybrid automata rely on multiple modules in conventional switched affine systems [17]. In contrast, our approach simplifies it to a single network with a window of past state-action as input.

Specifically, the hybrid automata are designed to output a one-hot latent, making this an unsupervised classification problem. We employ a combination of a softmax classifier and categorical sampling to determine the mode. First, the probability distribution over modes is computed as

$$p = f_{\texttt{softmax}} \circ f_{\texttt{DHA\_logit}}(o_{t-k:t}, a_{t-k-1:t-1}; \theta_{\texttt{DHA}}), \tag{4}$$

where p is the probability for each mode, $f_{\texttt{softmax}}$ is the softmax loss, $f_{\texttt{logit}}$ is the logit function, $\theta_{\texttt{DHA}}$ represents the parameters of the DHA neural network. Next, categorical sampling is applied to p to obtain a one-hot latent vector δ, which indicates the selected dynamics mode:

$$\delta \sim \texttt{Categorical}(p), \quad \sum_{i=1}^{n} \delta_i = 1, \quad \delta_i \in \{0, 1\}, \tag{5}$$

where $\delta_i = 1$ represents the mode is $i_t = i$.

With the hybrid automata, we build K corresponding dynamics β-VAEs for K modes. For each mode, the β-VAE encodes the historical trajectory and extracts the time-sequence feature into a latent representation, which will be decoded into the next partial state o_{t+1}. The latent representation in β-VAE encodes substantial dynamic information; hence, we use it in the controller. For the encoder, we adopt a one-dimensional CNN architecture due to its superior ability to capture temporal dependencies, which can be expressed by

$$z_t = \sum_{i=1}^{M} \delta_i \cdot f_{\text{enc}}^{i_t}(o_{t-k:t}, a_{t-k-1:t-1}; \theta_{\text{enc}}), \tag{6}$$

where $z_t \in \mathbb{R}^n$ is the latent representation, and the mode selector is included using δ. The decoder then uses this representation to predict o_{t+1}, c_{t+1}, which can be expressed by

$$\hat{o}_{t+1}, \hat{c}_{t+1} = \sum_{i=1}^{M} \delta_i \cdot f_{\text{dec}}^{i_t}(z_t; \theta_{\text{dec}}), \tag{7}$$

where $c_{t+1} \in [0, 1]^n$ is the probability that each leg is in contact at time $t+1$. We include contact information to make learning hybrid switching easier for the networks.

Typically, the ground truth for mode label and partial states is required to train these two neural networks [9]; however, the mode label is hard to obtain even in simulation. To address this, motivated by Mitchell et al. [50], we utilize the unsupervised learning method to train the mode selector. We train the mode selector and the β-VAE simultaneously as (3), where the mode is self-determined by constructing the loss function as

$$\mathcal{L}_{\texttt{vae}} = \sum_{t} \texttt{MSE}(\hat{o}_{t+1}, o_{t+1}) + \texttt{BCE}(\hat{c}_{t+1}, c_{t+1}) + \beta \mathcal{L}_{\mathcal{KL}}), \tag{8}$$

where o_{t+1}, c_{t+1} are the ground truth values, $\texttt{MSE}$ represents mean-square-error loss, $\texttt{BCE}$ represents binary-cross-entropy loss. The KL divergence prevents the encoder from fitting the data too flexibly. It encourages the distribution of the latent variable to be close to a unit Gaussian, while β is used to control the trade-off scale. The hybrid automata is trained by minimizing the prediction error of o_{t+1}, c_{t+1}, where the correct mode label will result in a lower prediction error. Such unsupervised learning eliminates the need for ground truth mode labels, which are often difficult to obtain in high-dimensional systems. The discrete categorical sample results in discontinuity of gradients, so we apply the straight-through-gradient method [51] to realize backpropagation during the training. Moreover, the gradient of the discrete-time automata is independent of PPO, ensuring accurate mode identification. However, the gradient backpropagates through the encoder to extract useful temporal features.

In addition, we encourage the mode to be distinguished by minimizing the information entropy of the mode probability p, resulting in the final loss:

$$\mathcal{L}_{\texttt{DHA}} = \mathcal{L}_{\texttt{vae}} + \mathcal{H}(p), \tag{9}$$

where $\mathcal{H}$ represents information entropy, aiming to ensure that the probabilities of modes are as distinct as possible and to prevent confusion between modes.

We adopt a discrete-time formulation compared to [9] that models hybrid dynamics using a continuous flow approach. This formulation unifies mode switching and dynamics within a single framework rather than treating them separately. Furthermore, unlike [9] that requires pre-segmentation of the trajectories to distinguish the mode, our method integrates mode selection directly into the training process, streamlining the entire workflow. The detailed network architecture and hyperparameters can be found in the Appendix.

B. Multi-Critic Reinforcement Learning

Inspired by Zargarbashi et al. [40], Xing et al. [52], we adopt the concept of multi-critic learning and apply it to the design of multi-task objectives for different motion phases. Designing an intuitive reward function for the robotic dog to perform skateboarding maneuvers is challenging. Therefore, we leverage sparse contact-based rewards to guide the robot in executing

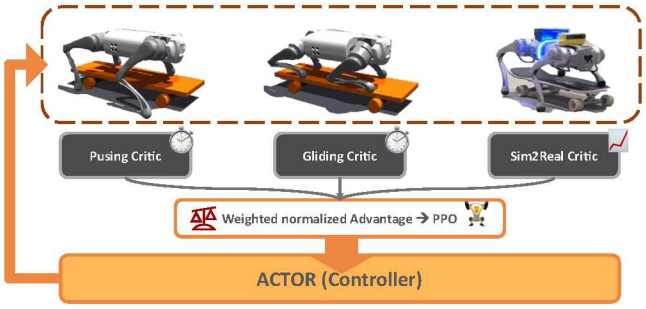

Fig. 5: Multi-Critic Skateboard Task

smooth pushing motions and gliding on the skateboard—two distinctly different movement patterns. Unlike the approach in Fu et al. [53], where minimizing energy consumption is prioritized, we found that this approach struggles to induce the gliding motion naturally. This difficulty arises because skateboarding, compared to standard quadrupedal locomotion, has a much smaller stability margin. As a result, the robot tends to avoid using the skateboard altogether to minimize energy loss rather than embracing it as part of the task.

To address this challenge, we design two distinct tasks corresponding to different phases of the cycle: *gliding* and *pushing*. During the pushing phase, the primary objective is to track the desired speed, while during the gliding phase, the goal is to maintain balance and glide smoothly on the skateboard. Although we specify the motion phase transitions using cyclic signals during training to ensure balanced learning of both tasks, in real-world deployment, these cyclic signals can be adapted on the fly or generated based on the velocity. The cyclic signal is an indicator that informs the robot of which movement pattern to adopt.

However, because the speed-tracking rewards and regularization terms for sim-to-real adaptation are dense, while the contact-related rewards are inherently sparse at the early stages of exploration, using a single critic to estimate the value of all rewards can lead to the advantages of sparse rewards being diluted by the dense rewards during normalization, which makes it difficult for the robot to learn the desired different style behaviors [40].

To mitigate this and generate different style motions, we introduce a multi-critic framework, as illustrated in Fig. 5. We define three reward groups, each associated with a different critic:

- **Gliding Critic**: Responsible for evaluating rewards related to gliding, such as speed tracking.
- **Pushing Critic**: Focused on sparse contact-related rewards during the pushing phase.
- **Sim2Real Critic**: Responsible for rewards related to sim-to-real adaptation, such as action regularization.

Each critic is updated separately to estimate its value function. When calculating the overall advantage, the outputs of three critics are normalized and combined using weighted summation. Similar to Zargarbashi et al. [40], advantage

TABLE II: Summary of Reward Terms and Their Expressions. Each term is multiplied by its phase coefficient (δ_{glide} or δ_{push}) if it belongs to a specific phase, and scaled by $w.$ as listed in the text.

Gliding Critic Reward	**Expression**
Feet on board	$\delta_{\text{glide}} \sum_{i=1}^{4} \Big(\lVert \mathbf{p}_{\text{feet},i} - \mathbf{p}_{\text{glide},i} \rVert < 0.05 \Big)$
Contact number	$\delta_{\text{glide}}\ R_{contact\_num}(22)$
Feet distance	$\delta_{\text{glide}} \exp\Big(- \sum_{i=1}^{4} \lVert \mathbf{p}_{\text{glide},j} - \mathbf{p}_{\text{feet},j} \rVert \Big)$
Joint positions	$\delta_{\text{glide}} \exp\Big(-\sum_{i=1}^{12} (q_i - q_i^{\text{glide}})^2 \Big)$
Hip positions	$\delta_{\text{glide}} \exp\Big(-\sum_{i \in Hip} (q_i - q_i^{\text{glide}})^2 \Big)$
Pushing Critic Reward	**Expression**
Tracking linear velocity	$\delta_{\text{push}} \exp\Big(-\frac{1}{\sigma} \lVert \mathbf{v}_x^{\text{cmd}} - \mathbf{v}_x \rVert^2 \Big)$
Tracking angular velocity	$\delta_{\text{push}} \exp\Big(-\frac{(\omega_z^{\text{cmd}} - \omega_z)^2}{\sigma_{\text{yaw}}} \Big)$
Hip positions	$\delta_{\text{push}} \exp\Big(-\sum_{i \in Hip} (q_i - q_i^{\text{push}})^2 \Big)$
Orientation	$\delta_{\text{push}} \lVert \mathbf{g}_{xy} \rVert^2$
Sim2Real Critic Reward	**Expression**
Wheel contact number	$\Big(\sum_{i \in \text{wheels}} c_i = 4 \Big)$
Board–body height	$\exp\Big(-4 \big\lvert (z_{\text{body}} - z_{\text{board}}) - 0.15 \big\rvert \Big)$
Joint acceleration	$\sum_{i=1}^{12} \left(\frac{\text{clip}\left(\dot{q}_i^{(t-1)} - \dot{q}_i^{(t)}, -10, 10\right)}{\Delta t} \right)^2$
Collisions	$\sum_{i \in \mathcal{P}} \Big(\lVert \mathbf{f}_i \rVert > 0.1 \Big)$
Action rate	$\lVert \mathbf{a}^{(t)} - \mathbf{a}^{(t-1)} \rVert$
Delta torques	$\lVert \boldsymbol{\tau}^{(t)} - \boldsymbol{\tau}^{(t-1)} \rVert^2$
Torques	$\lVert \boldsymbol{\tau} \rVert^2$
Linear velocity (z-axis)	$\lVert \text{clip}(v_z, -1.5, 1.5) \rVert^2$
Angular velocity (x/y)	$\lVert \text{clip}(\boldsymbol{\omega}_{xy}, -1, 1) \rVert^2$
Base orientation	$\lVert \mathbf{g}_{xy} \rVert^2$
Cycle Calculation	**Expression**
Cycle	T
Phase	$\phi \leftarrow \sin(2\pi\, t/T)$
Still Indicator	δ_{still}
Glide Indicator	$\delta_{\text{glide}}^{(t)} = LPF\Big([\phi < 0.5] \vee \delta_{\text{still}} \Big)$
Push Indicator	$\delta_{\text{push}}^{(t)} = LPF\Big([\phi \geq 0.5] \wedge \neg \delta_{\text{still}} \Big)$
Low pass filter	LPF

weights of each critic are treated as hyperparameters. This approach reduces the sensitivity to reward tuning, simplifying the reward design process. Specifically, the reward function is presented in the Table. II and their values are in Appendix A.

C. Beta Distribution Policy

In quadrupedal locomotion tasks, the mainstream frameworks typically assume a Gaussian distribution as the policy distribution in PPO due to its intuitive parameterization and ease of shape control [54]. However, when the action space has strict bounds (e.g., joint position limits to prevent collisions in quadrupedal robots), the Gaussian distribution can introduce bias in policy optimization by producing out-of-bound actions that need to be clipped [41]. While this issue has been noted in prior work, it has not been widely addressed in the context of robotic locomotion.

We observe that the Beta distribution offers a significant advantage over the Gaussian distribution in effectively utilizing the action space under sparse reward conditions. In contrast,

Gaussian policies may increase the variance excessively to explore more action space and trigger potential sparse rewards. This aggressive strategy can lead to suboptimal performance, getting stuck in local optima, and even result in hardware damage and safety hazards during real-world deployment. Therefore, we introduce the Beta distribution as the policy distribution for our framework.

In our implementation, the policy outputs the shape parameters (α, β) of the Beta distribution for each joint independently. To ensure that the Beta distribution remains unimodal $(\alpha > 1, \beta > 1)$, we modify the activation function as follows:

$$\text{SoftplusWithOffset}(x) = \log(1 + e^x) + 1 + 10^{-6}. \quad (10)$$

The standard Beta distribution is defined over $[0, 1]$:

$$f(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1 - x)^{\beta-1}, \quad \alpha, \beta > 0. \quad (11)$$

To adapt this to the robot action space $[-a_{\max}, a_{\max}]$ where $a_{\max} > 0$, we apply the following transformation:

$$a \sim \mathcal{B}(\alpha, \beta), \quad a' = a \cdot 2a_{\max} - a_{\max}. \quad (12)$$

In the expression, the a' denotes the actual action that the robot will execute. This approach enables the policy to produce bounded, valid actions within the desired range, preventing out-of-bound behaviors while entirely using the action space. We provide a proof in the Appendix B demonstrating that our method does not introduce bias or result in a variance explosion.

V. EXPERIMENTS

In the experiment, we answer the following questions regarding the performance in hybrid robotics systems:

- **Q1:** Why is a hybrid dynamics system better than Single Dynamics modeling?
- **Q2:** Can our method identify the skateboarding mode?
- **Q3:** Can our method achieve skateboarding in the real world with disturbances?

A. Prediction of Dynamics

To answer **Q1**, we compare the dynamics prediction loss between different maximal modes, corresponding to the dimension of the mode indicator $|\delta|$. Among these modes, the condition where the number of modes $|\delta|$ equals one represents using one network to model the whole dynamics, like [27]. We trained each condition three times with random seeds for network initialization. The curve shows the average reconstruction loss under different modes as shown in Fig. 7. The loss is the highest when the maximum number of modes is 1. After incorporating the hybrid dynamics idea, different modes are switched to guide the conversion and mutation of flow dynamics; the reconstruction loss is minor. Starting from the maximum number of modes 2, the improvement in prediction accuracy begins to plateau.

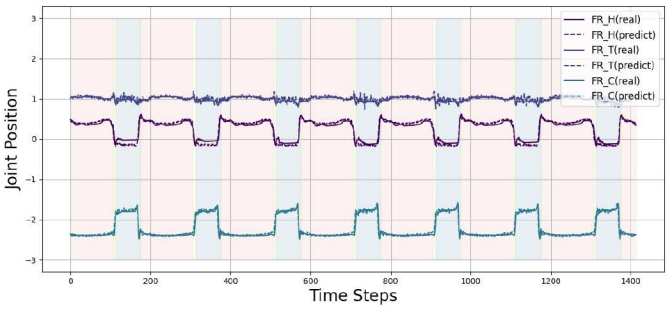

Fig. 6: Trajectory Prediction Visualization: The comparison between the actual position trajectory (solid line) and the predicted position trajectory (dashed line) for the right front leg joint motor of the physical Go1 robot during skateboarding is shown. We selected the right front leg joint, which exhibits the largest range of motion, as the visualization target. During deployment, the system utilizes the mode selection results from the automata to choose the corresponding decoder for prediction, consistent with the training process.

This is consistent with our assumption that building the system as a hybrid dynamics system in a system with mutation and other properties is more reasonable. Considering that the two primary states of the skateboard are the skateboard up and the skateboard down, when $|\delta| \geq 2$, there is a marginal effect of improvement. When $|\delta| \geq 4$, the prediction accuracy can hardly be improved and converges to the state of mode=3. **It is worth noting that we only set the maximum number of modes to 3, rather than requiring that all three modes must be present.** Therefore, in subsequent experiments, we believe that a mode count of three is the reasonable maximal number of modes for this system, corresponding to three motion modes: on the skateboard, pushing the skateboard under the skateboard, and being in the air between the two. We will showcase the mode identification in section V-B.

To validate the accuracy of the predicted dynamics, we deployed our DHAL algorithm on the physical robot and recorded both the predicted and actual dynamics in real time. In this experiment, we applied a clock signal with a fixed period of 2.5 seconds to the controller, alternating the upward and downward movements of the skateboard. Notably, this clock signal differs from the one used during the training phase (4s), providing an additional test for the accuracy of the predicted dynamics under new conditions. As shown in Fig. 6, the representation can accurately predict the trajectory and jumps of the states. In the next section, we will evaluate the capability of the method for mode identification and its ability to adjust phases on the fly.

B. Mode Identification

To answer **Q2**, we collected real-world trajectories of the robot skateboarding along with the modes selected by the controller for visualization, as shown in Fig. 8. Additionally, we used different RGB LED colors to represent the dynamic modules selected by the hybrid automata. The figure shows that when the robot is in the gliding phase, with all four feet standing firmly on the skateboard, the hybrid automata classify this state as mode 3 (red). When the right front foot starts lifting off the ground and entering the swing phase, the

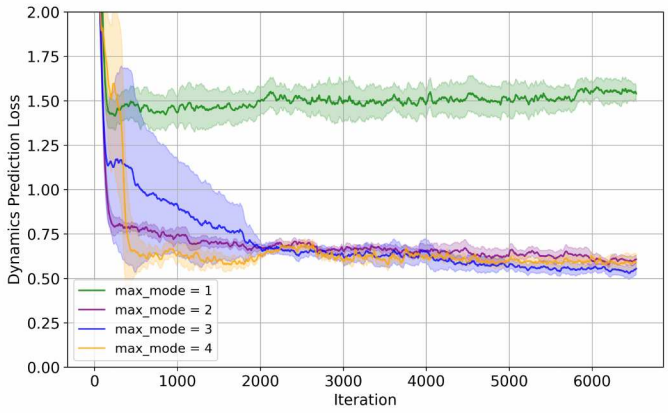

Fig. 7: Dynamics Prediction Loss: The dynamics prediction loss $\text{MSE}(\hat{o}_{t+1}, o_{t+1})$ during training is shown in the figure, where the thick line represents the average loss, and the shaded regions indicate the confidence intervals across different seeds.

automata naturally transition to mode 1 (green). This brief transition corresponds to the sharp change in joint angles shown in the figure. Once both the right legs are entirely in contact with the ground, the automata switch to mode 2 (blue).

This sequence of mode selections and transitions is smooth and explicitly aligns with the decomposition of skateboarding motion: (1) gliding phase, (2) airborne phase transitioning on and off the skateboard, and (3) pushing phase. These results demonstrate that our hybrid automata make mode selections highly consistent with physical intuition.

Additionally, we applied t-distributed stochastic neighbor embedding (t-SNE) to reduce the dimensionality of the hidden layer outputs from the controller. As illustrated in Fig. 9, the latent space exhibits a clearly defined distribution across different modes. Interestingly, the resulting latent structure is remarkably similar to that reported in [50]. Specifically, the red region primarily corresponds to the data collected during the movements on the skateboard, the blue region represents states where the right two legs are in contact with the ground during pushing, and the green region corresponds to the airborne phases when transitioning on and off the skateboard.

This observation highlights two key points: (1) our controller effectively handles motion control tasks across different modes, and (2) our DHAL module can distinctly and accurately differentiate between various modes.

C. Performance on Skateboarding

To answer **Q3**, this section presents ablation studies and real-world experiments to evaluate our method. The analysis is divided into the following parts: (1) Comparison of training returns, (2) Comparison between the single-critic method, and (3) Experiments in real-world scenarios with disturbances, including success rate statistics.

Comparison of training returns: Since the skateboarding task is designed to be contact-guided, the training process exhibits significant randomness, leading to considerable variance in training curves even for the same method. Therefore, the primary goals of this experiment are: (1) to identify the key factors driving successful training, and (2) to evaluate whether our method can approach optimal performance.

For a comparative evaluation, we compared the following algorithms with access to proprioception only:

1) **PPO-oracle-beta**: Training a policy with full privileged observations and the Beta distribution.
2) **DreamWaq** [27]: Training a dynamics module to estimate velocity and future observation.
3) **PPO-curiosity** [54, 10]: Training directly with only proprioception and following the curiosity reward design[10].

As shown in Fig. 10, our method could achieve comparable performance with **PPO-oracle-beta**, which has privileged observation about skateboard information. Notably, for the methods with the Gaussian distribution, the robot cannot learn how to do skateboarding, even leading to dangerous motion, which makes real-world deployment infeasible. The result aligns with the discussion in Section IV-C: in environments with high exploration difficulty, the Gaussian distribution tends to prioritize increasing variance to expand the exploration range, thereby randomly encountering "reward points". However, due to the physical constraints of the robot, this exploration strategy introduces bias, causing Gaussian distribution policies to favor movements closer to the constraints and ultimately leading to failure [41].

Comparison with single-critic: We trained our multi-critic method and the single-critic method for comparison. Since our multi-critic approach normalizes the advantages of different reward groups and combines them through weighted summation, while the single-critic approach lacks this weighting advantage mechanism, we evaluated two configurations of the single-critic method:

- **Single-critic-w-transfer**: A single-critic setup with the same reward configuration as the multi-critic method, but with new reward weights transferred based on the advantage weights
- **Single-critic-wo-transfer**: A single-critic setup with the same reward configuration and weights as the multi-critic method

As shown in Fig. 11, for the single-critic approach without transferred weights, the robot exhibited aggressive and erratic movements, making it challenging to handle disturbances. During forward motion, excessive hyperflexion at the foot caused it to get stuck, and when mounting the skateboard, the hind legs often slipped off. In contrast, the single-critic approach with transferred weights successfully mounted the skateboard. However, during the pushing phase, the robot primarily relied on its hind legs, leaving the front legs suspended for extended periods, resulting in an unnatural gliding posture.

With our multi-critic training scheme, the robot achieved a smooth and natural motion, efficiently executing rapid pushing and demonstrating significantly more stable and graceful transitions on and off the skateboard. We obtained results similar to Mysore et al. [55], proving that the multi-critic approach is well-suited for multi-style learning.

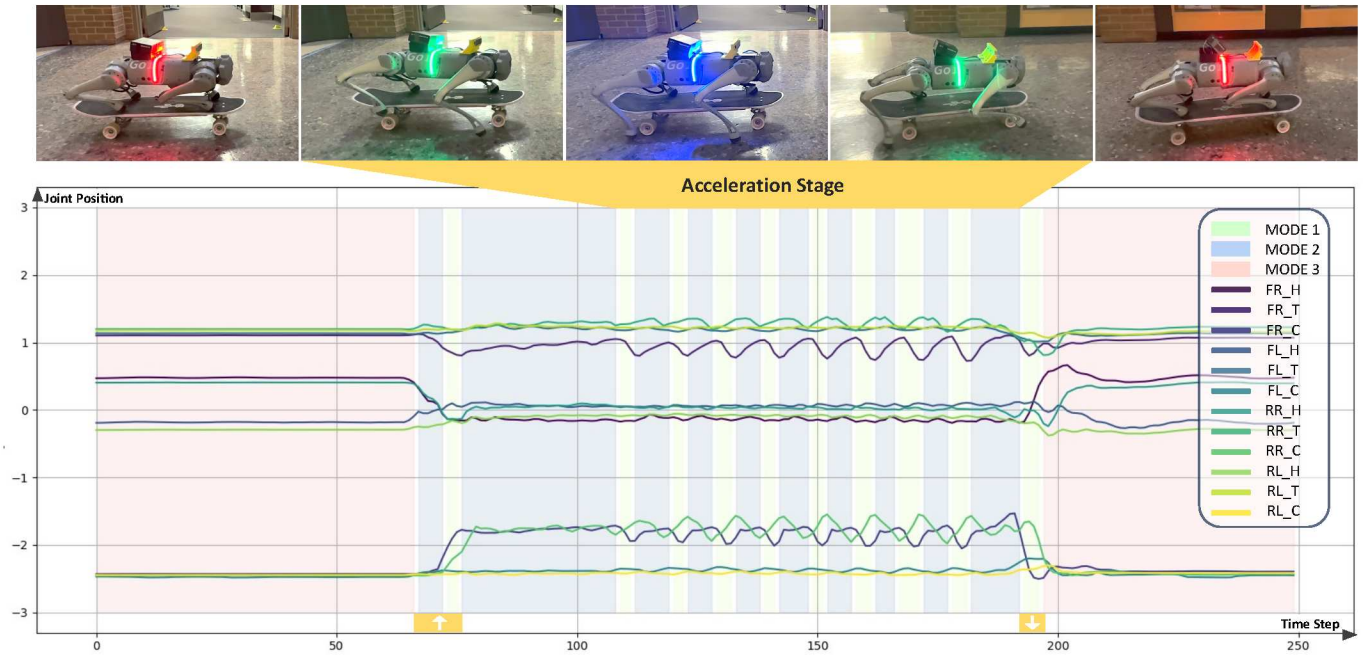

Fig. 8: Effectiveness of mode identification. In the real-world deployment, we light up different RGB light bar colors according to the mode to show the switching between different modes. The following figure shows the change in joint position relative to time in the test, and the different background colors represent different corresponding modes. [H, T, C] denote the Hip, Thigh, and Calf Joints, respectively.

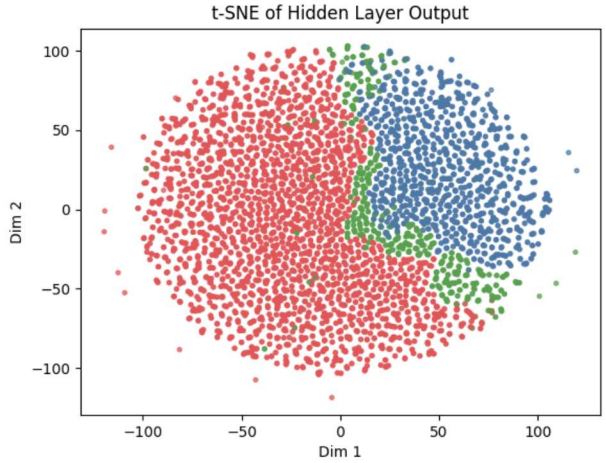

Fig. 9: Visualization of hidden layer of the controller: Scatter points in different colors correspond to the different modes identified by the system, consistent with Fig. 8. Specifically, [green, blue, red] represent [mode 1, mode 2, mode 3], respectively.

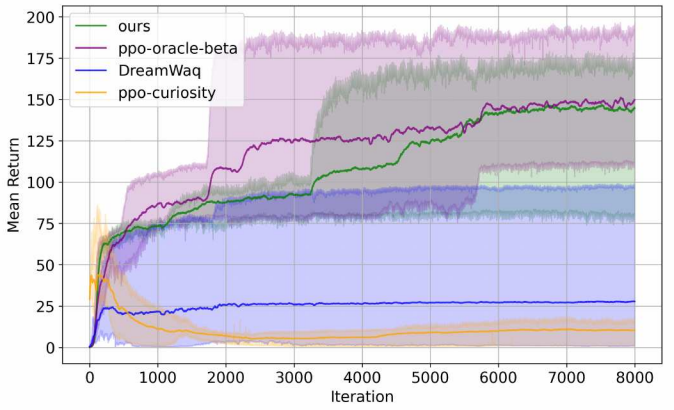

Fig. 10: Comparison of Training Rewards: Comparison of mean reward during training is shown in the figure, where the thick line represents the average return, and the shaded regions indicate the maximal and minimal reward across different seeds. Each method was trained using four random seeds to evaluate performance.

Quantitative Experiments: We conducted quantitative experiments in real-world scenarios to evaluate whether our method can complete the skateboarding task under real-world noise and disturbances. These disturbances include, but are not limited to, sensor noise, skateboard property variations, terrain irregularities, and dynamics noise [26]. Using the trained model, we tested the following scenarios: (1) smooth ceramic flooring, (2) soft carpeted flooring, (3) disturbance, (4) slope terrain, (5) single-step terrain, (6) uneven terrain. Each scenario was tested ten times, with each test containing at least one full cycle of mounting and dismounting the skateboard. The test terrain is shown in Fig. 1 and success rate statistics are shown in Table. III. More extreme terrain experiments and validation in another task could be found in Appendix E-D.

VI. Limitations and Discussion

1. **Perception Limitations**: To connect the robot's left front foot to the skateboard (only one foot), we assumed a spherical joint to prevent the skateboard from completely detaching from the robot. The transition from walking to skateboarding presents a significantly greater challenge, requiring hardware modifications, such as adjusting the camera layout

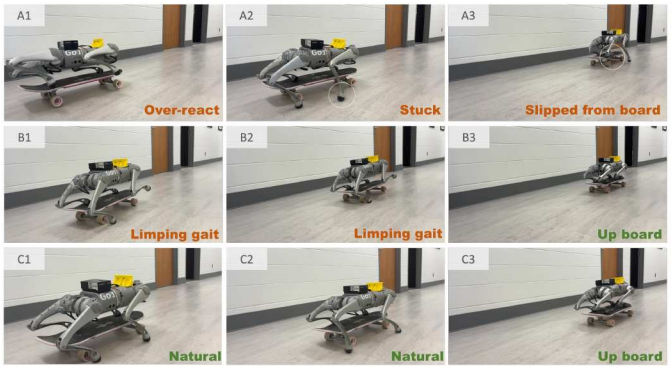

Fig. 11: Comparison between single-critic policy and multi-critic policy: Single-critic-wo-transfer ($A1 \sim A3$), Single-critic-w-transfer ($B1 \sim B3$), ours ($C1 \sim C3$).

TABLE III: Success Rate Comparison: We deployed each method on a real robot to evaluate the success rate. Each method was tested five times per scenario. Success was defined as completing at least one full up-board and down-board motion, traversing over a distance of more than 5 meters, and avoiding abrupt movements or detachment from the skateboard. ✗ indicates complete failure (Massive torque caused the joints protection state; for hardware protection, we first test the torque value in simulation to ensure it will not exceed the safety range).

Method	**Ceramic**	**Carpet**	**Disturbance**
Ours	**100%**	**100%**	**100%**
Our-wo-MC(transferred)	100%	100%	60%
Our-wo-MC	60%	60%	40%
Ours-wo-Beta	✗	✗	✗
DreamWaq[27]	✗	✗	✗
Method	**Slope**	**Single-step**	**Uneven**
Ours	**80%**	**100%**	**60%**
Our-wo-MC(transferred)	60%	40%	60%
Our-wo-MC	0%	40%	40%
Ours-wo-Beta	✗	✗	✗
DreamWaq[27]	✗	✗	✗

and incorporating multiple cameras to locate the skateboard. Furthermore, we did not consider obstacle avoidance during skateboarding using perception-based methods. Initially, we attempted to use the RealSense T265 for state estimation, but later determined it was unnecessary for this task. However, for future work, when the foot is not fixed to the board, the state estimation methods [56, 57, 58, 59, 60] need to be carefully integrated.

2. **Complex skill Generalization**: Our method cannot generalize to extreme skateboarding techniques equivalent to those of human athletes, such as performing an ollie. The current simulation setup cannot accurately replicate the motion and contact dynamics of passive wheels in such challenging scenarios. Instead, we relied on approximations and alternative techniques to simulate these dynamics as realistically as possible.

3. **Limitations in Dynamics Learning**: The learned dynamics are not yet precise enough for model-based control. Furthermore, the coupling between the controller and the dynamics predictor prevents iterative optimization, such as that used in MPC, limiting the flexibility and efficiency of our approach.

4. **Non-Trivial Environment Design:** The environment setting and design for robot skateboarding is non-trivial. This part requires manual design and inspection. We believe that in the future, integrating environment generation with large models [61] could potentially help address this challenge.

VII. Conclusion

We proposed the Discrete-time Hybrid Automata Learning (DHAL) framework to address mode-switching in hybrid dynamical systems without requiring trajectory segmentation or event function modeling. By combining a multi-critic architecture and a Beta distribution policy, our method demonstrates robust handling of contact-guided hybrid dynamics, as validated through the challenging task of quadrupedal robot skateboarding. Real-world experiments showed that our approach achieves smooth and intuitive mode transitions, effectively balancing gliding and pushing behaviors. While limitations remain, such as terrain generalization and coupling between the controller and dynamics predictor, DHAL offers a promising step toward learning-based control for hybrid systems in robotics.

Acknowledgments

M. Ghaffari was supported by AFOSR MURI FA9550-23-1-0400. We appreciate the valuable discussions, hardware guidance, and constructive feedback from Yulun Zhuang and Yi Cheng. We also extend our gratitude to Linqi Ye for the initial brainstorming and insightful suggestions.

Appendix A
Multi-Critic

A. Multi-Critic PPO Loss design

Motivated by Zargarbashi et al. [40], we define $\mathcal{P}$, $\mathcal{G}$, and $\mathcal{S}$ to represent the "Pushing," "Gliding," and "Sim2Real" tasks, respectively. Consequently, $r_{\mathcal{P}}, r_{\mathcal{G}}, r_{\mathcal{S}}$ denote the weighted sums of the specific reward groups, while $V_{\mathcal{P}}, V_{\mathcal{G}}, V_{\mathcal{S}}$ correspond to the respective value networks. The overall value loss is given by $L^{\text{value}} = L_{\mathcal{P}} + L_{\mathcal{G}} + L_{\mathcal{S}}$, where each term is defined as:

$$L_{\mathcal{P}} = \mathbb{E}_t \left[\left\| r_{\mathcal{P},t} + \gamma V_{\mathcal{P}}(s_{t+1}) - V_{\mathcal{P}}(s_t) \right\|^2 \right], \tag{13}$$

$$L_{\mathcal{G}} = \mathbb{E}_t \left[\left\| r_{\mathcal{G},t} + \gamma V_{\mathcal{G}}(s_{t+1}) - V_{\mathcal{G}}(s_t) \right\|^2 \right], \tag{14}$$

$$L_{\mathcal{S}} = \mathbb{E}_t \left[\left\| r_{\mathcal{S},t} + \gamma V_{\mathcal{S}}(s_{t+1}) - V_{\mathcal{S}}(s_t) \right\|^2 \right]. \tag{15}$$

Here, γ is the discount factor, and s_t represents the state at time t. Each value loss minimizes the temporal difference (TD) error of the corresponding reward group.

For advantage estimation in PPO, each reward group and its associated critic calculate the advantage separately based on the TD error. Taking "Pushing" as an example, the TD error is defined as:

$$\delta_{\mathcal{P},t} = r_{\mathcal{P},t} + \gamma(1 - d_t) V_{\mathcal{P},t+1} - V_{\mathcal{P},t}, \tag{16}$$

Fig. 12: Real-world Experiments in Skateboard Park. For additional demonstrations, please refer to our website, where more result videos are available.

where d_t is an indicator variable denoting whether the episode terminates at time t. The advantage is then calculated recursively as:

$$A_{\mathcal{P},t} = \delta_{\mathcal{P},t} + \gamma(1-d_t)\lambda A_{\mathcal{P},t+1}. \tag{17}$$

Here, λ is the Generalized Advantage Estimation (GAE) parameter, which balances the trade-off between bias and variance in advantage estimation. After calculating the advantage for "Pushing", it is normalized as follows:

$$\tilde{A}_{\mathcal{P},t} = \frac{A_{\mathcal{P},t} - \mu_{A_{\mathcal{P},t}}}{\sigma_{A_{\mathcal{P},t}} + \epsilon}, \tag{18}$$

where $\mu_{A_{\mathcal{P},t}}$ and $\sigma_{A_{\mathcal{P},t}}$ are the mean and standard deviation of the advantage values for the "Pushing" task, and ϵ is a small constant added for numerical stability. This process is repeated for both "Gliding" and "Sim2Real", resulting in normalized advantages $\tilde{A}_{\mathcal{G},t}$ and $\tilde{A}_{\mathcal{S},t}$. Finally, the weighted sum of all normalized advantages is computed as:

$$\tilde{A}_t = w_1 * \tilde{A}_{\mathcal{P},t} + w_2 * \tilde{A}_{\mathcal{G},t} + w_3 * \tilde{A}_{\mathcal{S},t}. \tag{19}$$

The surrogate loss is then calculated as in the standard PPO process:

$$L^{\text{surrogate}} = \mathbb{E}_t\left[\min\left(\alpha_t\tilde{A}_t,\ \text{clip}(\alpha_t, 1-\epsilon, 1+\epsilon)\tilde{A}_t\right)\right], \tag{20}$$

where α_t is the ratio between the new and old policy probabilities. Finally, the overall PPO loss is computed as:

$$L^{\text{PPO}} = L^{\text{value}} + L^{\text{surrogate}} - c\cdot\mathcal{H}(\pi_\theta). \tag{21}$$

Here, $\mathcal{H}(\pi_\theta)$ represents the entropy of the policy π_θ, encouraging exploration, and c is a weighting coefficient. This formulation integrates the value loss, the surrogate loss, and an entropy regularization term to achieve robust and efficient policy optimization.

B. Reward Detail

The contact number reward for the gliding phase is expressed as:

$$R = 2\cdot R_{skateb} - P_{ground}, \tag{22}$$

where R_{skateb} represents the reward for maintaining correct contact with the skateboard, and P_{ground} is the penalty for undesired ground contact.

The skateboard contact reward is given by:

$$R_{skateb} = \delta_{glide}\cdot(\sum_{i=1}^{4}\mathbb{1}(c_{skateb,i}) + 4\cdot\mathbb{1}(\sum_{i=1}^{4}\mathbb{1}(c_{skateb,i}) = 4)), \tag{23}$$

where δ_{glide} is a scaling coefficient for the gliding phase, $\mathbb{1}(c_{skateb,i})$ indicates whether the i-th foot is in contact with the skateboard, and an additional reward is provided if all four feet maintain contact.

The ground contact penalty is expressed as:

$$P_{ground} = \delta_{glide}\cdot(\sum_{i\in\{0,2\}}\mathbb{1}(\sim c_{skateb,i}) + \sum_{i\in\{0,2\}}\mathbb{1}(c_{ground,i})), \tag{24}$$

where $\mathbb{1}(\sim c_{skateb,i})$ penalizes the lack of skateboard contact for specific feet, and $\mathbb{1}(c_{ground,i})$ penalizes unintended ground contact.

This reward design encourages the agent to maintain stable contact with the skateboard while avoiding unnecessary ground contact, ensuring smooth and efficient gliding behavior. The detailed reward weights are shown in Table IV.

APPENDIX B
BETA DISTRIBUTION

A. Why Gaussian distribution policy introduces bias

Based on Chou et al. [41], when a Gaussian policy $\pi_\theta(a \mid s) = \mathcal{N}(\mu_\theta, \sigma_\theta^2)$ is employed in a bounded action space $[-h, h]$(whatever because of environment manually design or physical constraints of robots), any action a exceeding these

TABLE IV: Reward weights and Advantage weights for skateboarding environment design

Gliding Critic Reward	**Weight**
Feet on board	0.3
Contact number	0.3
Feet distance	1.8
Joint positions	1.2
Hip positions	1.2
Pushing Critic Reward	**Weight**
Tracking linear velocity	1.6
Tracking angular velocity	0.8
Hip positions	0.6
Orientation	-2
Sim2Real Critic Reward	**Weight**
Wheel contact number	0.8
Board–body height	1
Joint acceleration	-2.5e-7
Collisions	-1
Action rate	-0.22
Delta torques	-1.0e-7
Torques	-1.0e-5
Linear velocity (z-axis)	-0.1
Angular velocity (x/y)	-0.01
Base orientation	-25
Advantage	**Weight**
Gliding Critic Advantage	0.35
Pushing Critic Advantage	0.4
Sim2Real Critic Advantage	0.25

limits is clipped to $\text{clip}(a) \in [-h, h]$. Ideally, the policy gradient should be computed as

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\big[\nabla_\theta \log \pi_\theta(a \mid s)\, A^\pi(s,a)\big].$$

However, if the update uses the clipped action in the value function,

$$g_{\text{clip}} = \nabla_\theta \log \pi_\theta(\text{clip}(a) \mid s)\; A^\pi\big(s, \text{clip}(a)\big),$$

then integrating over all a reveals a discrepancy whenever $|a| > h$. Specifically,

$$\mathbb{E}[g_{\text{clip}}] - \nabla_\theta J(\pi_\theta) = \mathbb{E}_s\Bigg[\int_{|a|>h} \pi_\theta(a|s)(\nabla_\theta \log \pi_\theta(\pm h|s) A^\pi(s,\pm h) - \nabla_\theta \log \pi_\theta(a|s) A^\pi(s,a))\, da\Bigg] \neq 0.$$

Because a Gaussian often places non-negligible probability mass outside $\pm h$, this mismatch fails to cancel out, producing a biased gradient that nudges the policy to favor actions beyond the valid range. The Policy Gradient of the Gaussian distribution policy is shown below.

$$\nabla_{\sigma_\theta} J(\theta) = \frac{1}{\sigma_\theta^3}\mathbb{E}_{a\sim\pi_\theta}\left[\left((a-\mu_\theta)^2 - \sigma_\theta^2\right) A^\pi(s,a)\right].$$

Moreover, bias in the policy may tend to produce actions outside the valid range. Increasing the variance becomes a direct consequence of this tendency. This forms a positive feedback loop: the more the policy variance grows, the more out-of-bound actions are sampled, and the larger $(a-\mu_\theta)^2$ becomes in the gradient, even though the actions are physically clipped. Unlike the Beta policy, a Gaussian distribution can increase μ indefinitely. That is why the Gaussian policy needs a more cautious reward design.

B. Why Beta distribution does not introduce bias

For Beta distribution, we first need to rescale it from $[0,1]$ to $[-h,+h]$. Suppose $\tilde{z} \sim \text{Beta}(\alpha_\theta(s), \beta_\theta(s))$, which is supported on $[0,1]$ and define $a = \phi(\tilde{z}) = 2h(\tilde{z} - \frac{1}{2})$. Thus $a \in [-h,h]$.

Because ϕ is a smooth, bijective function from $[0,1] \to [-h,h]$, we can define the policy pdf as:

$$\pi_\theta(a \mid s) = \text{Beta}_{\alpha_\theta(s),\beta_\theta(s)}\left(\phi^{-1}(a)\right) \cdot \left|\frac{d}{da}\phi^{-1}(a)\right|.$$

Concretely,

$$\phi^{-1}(a) = \frac{a+h}{2h}, \quad \left|\frac{d}{da}\phi^{-1}(a)\right| = \frac{1}{2h}.$$

Hence,

$$\pi_\theta(a \mid s) = \text{Beta}_{\alpha_\theta,\beta_\theta}\left(\frac{a+h}{2h}\right)\cdot\frac{1}{2h}, \quad a \in [-h,h].$$

Let us verify the critical zero-integral property for the Beta policy:

$$\int_{-h}^{+h} \pi_\theta(a \mid s)\nabla_\theta \log \pi_\theta(a \mid s)\, da = \int_{-h}^{+h} \nabla_\theta \pi_\theta(a \mid s)\, da.$$

But

$$\int_{-h}^{+h} \pi_\theta(a \mid s)\, da = 1 \quad \text{(all the mass is inside } [-h,h]).$$

Thus,

$$\nabla_\theta \int_{-h}^{+h} \pi_\theta(a \mid s)\, da = \nabla_\theta[1] = 0.$$

Hence,

$$\int_{-h}^{+h} \nabla_\theta \pi_\theta(a \mid s)\, da = 0,$$

i.e.,

$$\int_{-h}^{+h} \pi_\theta(a \mid s)\nabla_\theta \log \pi_\theta(a \mid s)\, da = 0.$$

No boundary terms appear, because $\pi_\theta(a \mid s)$ is zero outside $[-h,h]$.Thus, if plug π_θ from above into the standard policy gradient formula (1), we could get an unbiased estimator:

$$\mathbb{E}\big[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s,a)\big] = \nabla_\theta \int_{-h}^{+h} \pi_\theta(a \mid s) Q^\pi(s,a)\, da,$$

which equals to $\nabla_\theta J(\pi_\theta)$. A $Beta(\alpha,\beta)$ distribution on $[0,1]$ has a well-known finite variance:

$$\text{Var}(Z) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}.$$

After rescaling the beta range for action $[-h,h]$,

TABLE V: Network Architecture and Training Hyperparameter

Network Hyperparameters	**value**
DHA Architecture	MLP
DHA Hidden Dims	[256, 64, 32]
VAE Encoder Architecture	1-D CNN
VAE Encoder time steps	20
VAE Encoder Convolutional Layers	Input channel = [30, 20]
VAE Encoder Convolutional Layers	Kernel=(6,4), Stride=(2,2)
VAE Decoder Hidden Dims	[256, 128, 64]
VAE Latent Dims	20
VAE KL Divergence Weight(β)	1e-2
Actor Hidden Dims	[512, 256, 128]
Gliding Critic Hidden Dims	[512, 256, 128]
Pushing Critic Hidden Dims	[512, 256, 128]
Sim2Real Critic Hidden Dims	[512, 256, 128]
PPO HyperParameters	**Weight**
Environments	4096
Collection Steps	24
Discount Factor	0.99
GAE Parameter	0.9
Target KL Divergence	0.1
Learning Rate Schedule	adaptive
Number of Mini-batches	4
Clipping Paramete	0.2

$$\mathrm{Var}(A) = 4h^2\,\mathrm{Var}(Z) \le 4h^2 \cdot \max_{Z\in\mathrm{Beta}} \mathrm{Var}(Z).$$

And we already know $\mathrm{Var}(Z) \le \frac{1}{12}$ (if $\alpha = \beta = 1$, uniform). So:

$$\mathrm{Var}(A) \le \frac{h^2}{3}.$$

The variance of a rescaled Beta cannot exceed $h^3/3$, for all $\alpha, \beta > 1$ (we assume the control policy should be unimodal).

C. Realization Detail

We assume the control policy for the locomotion scenario is unimodal; therefore, $\alpha, \beta > 1$. We define the activation function for the output layer of the actor as shown below:

$$\mathrm{SoftplusWithOffset}(x) = \log(1 + e^x) + 1 + 10^{-6}.$$

During training, the action is sampled from a beta distribution and scaled to $[-h, h]$. For deployment, we directly use the mean of distribution $\alpha/(\alpha + \beta)$ as the output.

APPENDIX C
NETWORK ARCHITECTURE AND TRAINING HYPERPARAMETER

In Table VI, we outline the hyperparameters for DHAL. Notably, the DHA is decoupled from both the encoder and the actor when PPO loss propagation and is only updated using the dynamics loss. During the PPO update, the PPO loss backpropagates through the actor and the encoder.

TABLE VI: Randomization and Noise

Property Randomization	**value**
Friction	[0.6, 2.]
Added Mass	[0, 3]kg
Added COM	[-0.2, 0.2]
Push robot	0.5m/s per 8s
Delay	[0, 20]ms
Sensor Noise	**Weight**
Euler Angle	$\mathcal{N} * 0.08$
Angular Velocity	$\mathcal{N} * 0.4$
Projected Gravity	$\mathcal{N} * 0.05$
Joint Position	$\mathcal{N} * 0.05$
Joint Velocity	$\mathcal{N} * 0.1$

APPENDIX D
ENVIRONMENT SETTING AND SIM2REAL DETAIL

A. Rolling Friction

We observed that in Isaac Gym, the simulation of rolling objects, particularly wheels, is not highly accurate. This is primarily reflected in the discrepancies between simulated and real-world rolling friction, as well as imprecise collision detection. To address this, we applied a compensation force during training to roughly approximate the effects of rolling friction on different terrains for the forward and backward motion of the skateboard. The following formula defines the compensation force:

$$F_{\text{push},x} = \begin{cases} F_{\text{rand}}, & \text{if } v_{\text{skate},x} > 0.3 \\ -F_{\text{rand}}, & \text{if } v_{\text{skate},x} < -0.3 \\ 0, & \text{otherwise} \end{cases}$$

$$F_{\text{rand}} \sim U(10, 25)$$

$$v_{\text{skate},x} = \text{quat\_rotate\_inverse}\,(q_{\text{skate}}, v_{\text{skate}})$$

Where$F_{\text{push},x}$ is the applied push force along the x-axis, $F_{\text{rand}} \sim U(10, 25)$ is the randomly sampled force, and $v_{\text{skate},x}$ is the skateboard velocity in its local frame, computed using the inverse quaternion rotation $\text{quat\_rotate\_inverse}(q_{\text{skate}}, v_{\text{skate}})$.

B. Skateboard Truck model

To simulate a realistic skateboard, we incorporated a bridge structure into the robotic skateboard, consisting of a front and rear bridge. Each bridge is modeled using a position-based PD controller to emulate spring dynamics, with the desired position and velocity set to zero at all times.

C. Contact Detection

We found that in Isaac Gym, the collision calculations for skateboard motion are imprecise. This is evident in the inaccuracies of the collision forces for passive rolling wheels as well as the collision forces between the robot and the skateboard. To address this, we designed the reward function to combine relative position error with collision forces to determine whether contact has occurred. This logic requires manual design and implementation.

TABLE VII: Reward weights for two single-critic method(w-transfer/ wo-transfer)

Gliding Critic Reward	**Weight**	**Weight(Transfer)**
Feet on board	0.3	$0.35 * 0.3$
Contact number	0.3	$0.35 * 0.3$
Feet distance	1.8	$0.35 * 1.8$
Joint positions	1.2	$0.35 * 1.2$
Hip positions	1.2	$0.35 * 1.2$
Pushing Critic Reward	**Weight**	**Weight(Transfer)**
Tracking linear velocity	1.6	$0.4 * 1.6$
Tracking angular velocity	0.8	$0.4 * 0.8$
Hip positions	0.6	$0.4 * 0.6$
Orientation	-2	$0.4 * -2$
Sim2Real Critic Reward	**Weight**	**Weight(Transfer)**
Wheel contact number	0.8	$0.25 * 0.8$
Board–body height	1	$0.25 * 1$
Joint acceleration	-2.5e-7	$0.25 * -2.5e-7$
Collisions	-1	$0.25 * -1$
Action rate	-0.22	$0.25 * -0.22$
Delta torques	-1.0e-7	$0.25 * -1.0e-7$
Torques	-1.0e-5	$0.25 * -1.0e-5$
Linear velocity (z-axis)	-0.1	$0.25 * -0.1$
Angular velocity (x/y)	-0.01	$0.25 * -0.01$
Base orientation	-25	$0.25 * -25$

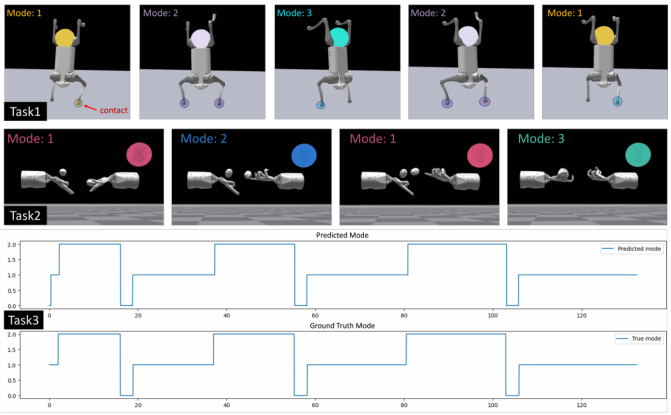

Fig. 13: Validation on other hybrid tasks and mode Identification.

APPENDIX E
EXPERIMENTAL SUPPLEMENTARY NOTES

A. Training Cost

We train policy on an NVIDIA RTX 3090; each iteration takes $3 \sim 4$ sec. Training a policy deployable in the real world costs $6 \sim 7$ hours.

B. Terrain setting

We trained our policy on flat ground, yet it demonstrated the ability to generalize to challenging terrains such as skateboard parks in real-world scenarios. At first, we also tried training it on complex terrain by including stairs and slopes. We observed no obvious advantage, while the training time increased. Therefore, the policies are all trained on flat ground.

C. Single-Critic Reward

In the experimental section, we compared the performance of single-critic and multi-critic approaches. To help readers understand the differences between the two single-critic setups, we list their reward configurations in the Table. VII.

D. Extreme Terrain Experiments

As shown in Fig. 12, we tested our robot in a skatepark specifically designed for extreme skateboarding, featuring challenging multi-level stair sets, U-shaped bowls, and terrain with cliff-like characteristics. Surprisingly, our algorithm remained relatively stable across these complex terrains. Although the robot occasionally deviated from the skateboard due to terrain disturbances, it could still recover and maintain a graceful skating posture.

E. Hybrid Dynamics automata Validation in other tasks

We apply the proposed framework to more tasks to demonstrate its effectiveness. **1) Robot Hand-stand Locomotion, 2) Dexterous Manipulation Hand-over, 3) Switching Linear Dynamical System** shown in Fig. 13. (1-2) aim at contact-rich problems, and 3) is a classic textbook example in hybrid dynamical systems with verifiable ground truth. In 1), we can identify the gait or contact mode. In 2), we can identify different stages of the object-handover task: in the air, pushed by the right hand, and caught by the left hand. In 3), we can compare our identification with the analytical solution.

Switching Linear Dynamical System (SLDS) is a hybrid system with multiple linear continuous systems. We consider the following SLDS [8]:

$$\frac{dx}{dt} = \begin{cases} xA + \begin{bmatrix} 0 & 2 \end{bmatrix} & \text{if } x_1 \geq 2 \\ \begin{bmatrix} -1 & -1 \end{bmatrix} & \text{if } x_0 \geq 0 \text{ and } x_1 < 2 \\ \begin{bmatrix} 1 & -1 \end{bmatrix} & \text{if } x_0 < 0 \text{ and } x_1 < 2 \end{cases}, \quad (25)$$

where $x = [x_0, x_1] \in \mathbb{R}^2$ and

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \quad (26)$$

This system will switch between three linear systems when the state reaches the guard.

REFERENCES

[1] Michael S Branicky. Introduction to hybrid systems. In *Handbook of networked and embedded control systems*, pages 91–116. Springer, 2005.

[2] Rajeev Alur, Thao Dang, and Franjo Ivančić1. Counter-example guided predicate abstraction of hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 208–223. Springer, 2003.

[3] Jonathan Touboul and Romain Brette. Spiking dynamics of bidimensional integrate-and-fire neurons. *SIAM Journal on Applied Dynamical Systems*, 8(4):1462–1506, 2009.

[4] E.R. Westervelt, J.W. Grizzle, and D.E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(1):42–56, 2003. doi: 10.1109/TAC.2002.806653.

[5] Koushil Sreenath, Hae-Won Park, and Ioannis Poulakakis. A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on mabel. *The International Journal of Robotics Research*, 30(9):1170–1193, 2011.

[6] Mostafa Khazaee, Majid Sadedel, and Atoosa Davarpanah. Behavior-based navigation of an autonomous hexapod robot

using a hybrid automaton. *Journal of Intelligent & Robotic Systems*, 102(2):29, 2021.

[7] Daniel L. Ly and Hod Lipson. Learning symbolic representations of hybrid dynamical systems. *Journal of Machine Learning Research*, 13(115):3585–3618, 2012. URL http://jmlr.org/papers/v13/ly12a.html.

[8] Ricky TQ Chen, Brandon Amos, and Maximilian Nickel. Learning neural event functions for ordinary differential equations. In *International Conference on Learning Representations*.

[9] Michael Poli, Stefano Massaroli, Luca Scimeca, Sanghyuk Chun, Seong Joon Oh, Atsushi Yamashita, Hajime Asama, Jinkyoo Park, and Animesh Garg. Neural hybrid automata: Learning dynamics with multiple modes and stochastic transitions. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9977–9989. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/5291822d0636dc429e80e953c58b6a76-Paper.pdf.

[10] Chong Zhang, Wenli Xiao, Tairan He, and Guanya Shi. Wococo: Learning whole-body humanoid control with sequential contacts. *arXiv preprint arXiv:2406.06005*, 2024.

[11] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2811–2817. IEEE, 2021.

[12] Yandong Ji, Zhongyu Li, Yinan Sun, Xue Bin Peng, Sergey Levine, Glen Berseth, and Koushil 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)*, pages 1479–1486. IEEE, 2022.

[13] Sangli Teng, William Clark, Anthony Bloch, Ram Vasudevan, and Maani Ghaffari. Lie algebraic cost function design for control on lie groups. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 1867–1874. IEEE, 2022.

[14] Sangli Teng, Ashkan Jasour, Ram Vasudevan, and Maani Ghaffari. Convex geometric motion planning of multi-body systems on lie groups via variational integrators and sparse moment relaxation. *The International Journal of Robotics Research*, page 02783649241296160, 2024.

[15] Sangli Teng, Ashkan Jasour, Ram Vasudevan, and Maani Ghaffari Jadidi. Convex Geometric Motion Planning on Lie Groups via Moment Relaxation. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.058.

[16] Thomas Corbères, Carlos Mastalli, Wolfgang Merkt, Ioannis Havoutis, Maurice Fallon, Nicolas Mansard, Thomas Flayols, Sethu Vijayakumar, and Steve Tonneau. Perceptive locomotion through whole-body mpc and optimal region selection, 2024. URL https://arxiv.org/abs/2305.08926.

[17] Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. Perceptive locomotion through nonlinear model-predictive control. *IEEE Transactions on Robotics*, 39 (5):3402–3421, 2023. doi: 10.1109/TRO.2023.3275384.

[18] Sangli Teng, Dianhao Chen, William Clark, and Maani Ghaffari. An error-state model predictive control on connected matrix lie groups for legged robot control. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8850–8857. IEEE, 2022.

[19] Sangli Teng, Yukai Gong, Jessy W Grizzle, and Maani Ghaffari. Toward safety-aware informative motion planning for legged robots. *arXiv preprint arXiv:2103.14252*, 2021.

[20] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43(4):572–587, 2024.

[21] Srinath Mahankali, Chi-Chang Lee, Gabriel B. Margolis, Zhang-Wei Hong, and Pulkit Agrawal. Maximizing quadruped velocity by minimizing energy. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11467–11473, 2024. doi: 10.1109/ICRA57147.2024.10609983.

[22] Tairan He, Chong Zhang, Wenli Xiao, Guanqi He, Changliu Liu, and Guanya Shi. Agile but safe: Learning collision-free high-speed legged locomotion. In *Robotics: Science and Systems (RSS)*, 2024.

[23] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *The International Journal of Robotics Research*, page 02783649241285161.

[24] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450. IEEE, 2024.

[25] Yi Cheng, Hang Liu, Guoping Pan, Houde Liu, and Linqi Ye. Quadruped robot traversing 3d complex environments with limited perception. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9074–9081. IEEE, 2024.

[26] Xinyang Gu, Yen-Jen Wang, Xiang Zhu, Chengming Shi, Yanjiang Guo, Yichen Liu, and Jianyu Chen. Advancing humanoid locomotion: Mastering challenging terrains with denoising world model learning. *arXiv preprint arXiv:2408.14472*, 2024.

[27] I Made Aswin Nahrendra, Byeongho Yu, and Hyun Myung. Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5078–5084, 2023. doi: 10.1109/ICRA48891.2023.10161144.

[28] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023.

[29] Minghuan Liu, Zixuan Chen, Xuxin Cheng, Yandong Ji, Rizhao Qiu, Ruihan Yang, and Xiaolong Wang. Visual whole-body control for legged loco-manipulation. *The 8th Conference on Robot Learning*, 2024.

[30] Yandong Ji, Gabriel B Margolis, and Pulkit Agrawal. Dribblebot: Dynamic legged manipulation in the wild. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5155–5162. IEEE, 2023.

[31] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[32] Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Conference on Robot Learning*, pages 22–31. PMLR, 2023.

[33] Yuxiang Yang, Tingnan Zhang, Erwin Coumans, Jie Tan, and Byron Boots. Fast and efficient locomotion via learned gait transitions. In *Conference on robot learning*, pages 773–783. PMLR, 2022.

[34] Gijeong Kim, Yong-Hoon Lee, and Hae-Won Park. A learning framework for diverse legged robot locomotion using barrier-based style rewards. *arXiv preprint arXiv:2409.15780*, 2024.

[35] Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing, 2024. URL https://arxiv.org/abs/2409.15610.

[36] Wanxin Jin. Complementarity-free multi-contact modeling and optimization for dexterous manipulation. *arXiv preprint*

arXiv:2408.07855, 2024.

[37] Tao Pang, HJ Terry Suh, Lujie Yang, and Russ Tedrake. Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models. *IEEE Transactions on robotics*, 39(6):4691–4711, 2023.

[38] Gijeong Kim, Dongyun Kang, Joon-Ha Kim, Seungwoo Hong, and Hae-Won Park. Contact-implicit model predictive control: Controlling diverse quadruped motions without pre-planned contact modes or trajectories. *The International Journal of Robotics Research*, 0(0):02783649241273645, 0. doi: 10.1177/02783649241273645. URL https://doi.org/10.1177/02783649241273645.

[39] Wanxin Jin and Michael Posa. Task-driven hybrid model reduction for dexterous manipulation. *IEEE Transactions on Robotics*, 2024.

[40] Fatemeh Zargarbashi, Jin Cheng, Dongho Kang, Robert Sumner, and Stelian Coros. Robotkeyframing: Learning locomotion with high-level objectives via mixture of dense and sparse rewards. *arXiv preprint arXiv:2407.11562*, 2024.

[41] Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *International conference on machine learning*, pages 834–843. PMLR, 2017.

[42] Anxing Xiao, Wenzhe Tong, Lizhi Yang, Jun Zeng, Zhongyu Li, and Koushil Sreenath. Robotic guide dog: Leading a human with leash-guided hybrid physical interaction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11470–11476. IEEE, 2021.

[43] Sangli Teng, Kaito Iwasaki, William Clark, Xihang Yu, Anthony Bloch, Ram Vasudevan, and Maani Ghaffari. A generalized metriplectic system via free energy and system identification via bilevel convex optimization. *arXiv preprint arXiv:2410.06233*, 2024.

[44] Rafal Goebel, Ricardo G Sanfelice, and Andrew R Teel. Hybrid dynamical systems. *IEEE control systems magazine*, 29(2):28–93, 2009.

[45] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017. Chapter 16.3.

[46] Shuxiao Chen, Jonathan Rogers, Bike Zhang, and Koushil Sreenath. Feedback control for autonomous riding of hovershoes by a cassie bipedal robot. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE, 2019.

[47] Ziang Liu, Genggeng Zhou, Jeff He, Tobia Marcucci, Li Fei-Fei, Jiajun Wu, and Yunzhu Li. Model-based control with sparse neural dynamics. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ymBG2xs9Zf.

[48] Haonan Chen, Yilong Niu, Kaiwen Hong, Shuijing Liu, Yixuan Wang, Yunzhu Li, and Katherine Rose Driggs-Campbell. Predicting object interactions with behavior primitives: An application in stowing tasks. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=VH6WIPF4Sj.

[49] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. *arXiv preprint arXiv:2306.14447*, 2023.

[50] Alexander Luis Mitchell, Wolfgang Xaver Merkt, Mathieu Geisert, Siddhant Gangapurwala, Martin Engelcke, Oiwi Parker Jones, Ioannis Havoutis, and Ingmar Posner. Vae-loco: Versatile quadruped locomotion by learning a disentangled gait representation. *IEEE Transactions on Robotics*, 39(5):3805–3820, 2023. doi: 10.1109/TRO.2023.3297015.

[51] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[52] Jiaxu Xing, Ismail Geles, Yunlong Song, Elie Aljalbout, and Davide Scaramuzza. Multi-task reinforcement learning for quadrotors. *IEEE Robotics and Automation Letters*, 2024.

[53] Zipeng Fu, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Minimizing energy consumption leads to the emergence of gaits in legged robots. In *5th Annual Conference on Robot Learning*.

[54] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[55] Siddharth Mysore, George Cheng, Yunqi Zhao, Kate Saenko, and Meng Wu. Multi-critic actor learning: Teaching rl policies to act with style. In *International Conference on Learning Representations*, 2022.

[56] Sangli Teng, Mark Wilfried Mueller, and Koushil Sreenath. Legged robot state estimation in slippery environments using invariant extended kalman filter with velocity update. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3104–3110. IEEE, 2021.

[57] Xihang Yu, Sangli Teng, Theodor Chakhachiro, Wenzhe Tong, Tingjun Li, Tzu-Yuan Lin, Sarah Koehler, Manuel Ahumada, Jeffrey M Walls, and Maani Ghaffari. Fully proprioceptive slip-velocity-aware state estimation for mobile robots via invariant kalman filtering and disturbance observer. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8096–8103. IEEE, 2023.

[58] Zijian He, Sangli Teng, Tzu-Yuan Lin, Maani Ghaffari, and Yan Gu. Legged robot state estimation within non-inertial environments. *arXiv preprint arXiv:2403.16252*, 2024.

[59] Sangli Teng, Harry Zhang, David Jin, Ashkan Jasour, Maani Ghaffari, and Luca Carlone. GMKF: Generalized moment kalman filter for polynomial systems with arbitrary noise. *arXiv preprint arXiv:2403.04712*, 2024.

[60] Maani Ghaffari, Ray Zhang, Minghan Zhu, Chien Erh Lin, Tzu-Yuan Lin, Sangli Teng, Tingjun Li, Tianyi Liu, and Jingwei Song. Progress in symmetry preserving robot perception and control through geometry and learning. *Frontiers in Robotics and AI*, 9:969380, 2022.

[61] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.