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

Expert Composer Policy: Scalable Skill Repertoire for Quadruped Robots

Guilherme Christmann*, Ying-Sheng Luo*, Wei-Chao Chen
Inventec Corporation, Taipei, Taiwan
{*guilherme.christmann, luo.ying-sheng, chen.wei-chao*}*@inventec.com*

***Abstract*— We propose the expert composer policy, a framework to reliably expand the skill repertoire of quadruped agents. The composer policy links pair of experts via transitions to a sampled target state, allowing experts to be composed sequentially. Each expert specializes in a single skill, such as a locomotion gait or a jumping motion. Instead of a hierarchical or mixture-of-experts architecture, we train a single composer policy in an independent process that is not conditioned on the other expert policies. By reusing the same composer policy, our approach enables adding new experts without affecting existing ones, enabling incremental repertoire expansion and preserving original motion quality. We measured the transition success rate of 72 transition pairs and achieved an average success rate of 99.99%, which is over 10% higher than the baseline random approach, and outperforms other state-of-the-art methods. Using domain randomization during training we ensure a successful transfer to the real world, where we achieve an average transition success rate of 97.22% (N=360) in our experiments.**

I. Introduction

As the robotics community refines, improves, and develops new skills for robots, an issue arises: how to enable access to a vast repertoire of skills? Different tasks commonly require different controllers regarding inputs, learning signals, such model architectures, such that switching from one controller to another can be unstable or impossible. The problem is further magnified in dynamic locomotion controllers. Focusing on real-world deployment, we wish to keep existing skills intact while incorporating new abilities that interconnect with existing ones, creating an ever-expanding library of skills.

A common approach to handle multiple skills is through a mixture of experts and hierarchical controllers. Low-level experts are trained in a single skill and integrated into a coherent controller using a higher-level policy, through gating modules that mixes actions of low-level experts with additive [1], [2], or multiplicative composition [3]. However, adding a new low-level expert requires re-training the controllers, which inevitably affects the behavior of the low-level experts, degrading their quality and limiting the ability to scale the number of skills robustly [4].

Rather than mixing the experts, we propose keeping them independent and using a mechanism to compose the skills over time sequentially. This goal of preserving the quality and behavior of low-level experts is prevalent in character controllers used in animation [5], [6]. A similar premise exists in physics-simulated characters [4], [7], where transition motions derive from the careful selection of motion parameters such as timing and agent poses. However, these approaches execute transitions between controllers through an instantaneous switch mechanism, limiting their applicability for linking skills with distinctive motions.

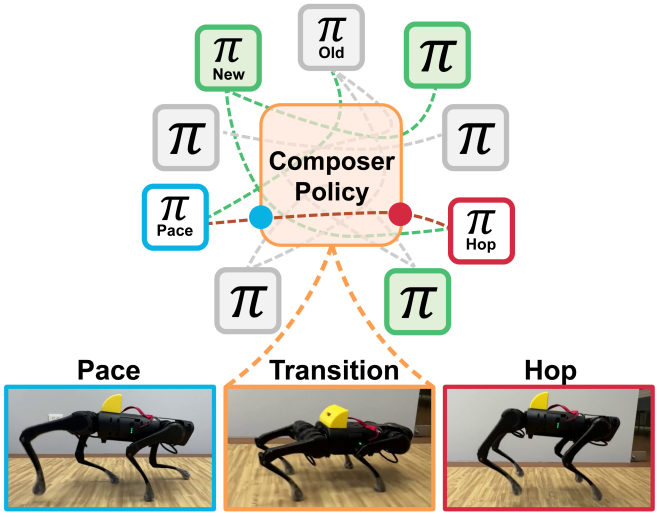

Fig. 1. Expert composer policy generates novel transitions between arbitrary agent states, enabling skill repertoire expansion while preserving the motion quality of the original experts.

In this work, we propose a framework that enables a large repertoire of skills, via smooth and dynamic transition trajectories. This is line with works in skill chaining and policy sequencing [8]–[10]. We train individual skills with dedicated policy controllers, further referred to as experts. These experts are trained in a physics-enabled simulation environment with randomized physical properties to enable transfer to real-world robots. Next, we introduce our expert composer policy that generates transition trajectories between arbitrary agent states, bridging every pair of experts smoothly. Inspired by [11], we employ tolerance bounds on physical properties that shrink over time to generate sensible trajectories. With a large skill repertoire of experts and the composer policy, we can create a coherent controller by switching to the composer policy between the execution of two experts (Figure 1). With our method, adding new experts does not require additional training processes, allowing us to

*These authors contributed equally, listed alphabetically by last name.

maintain all previously learned skills while accommodating new ones. Our list of contributions is as follows:

- A novel transition mechanism that generates smooth trajectories between arbitrary states of physics-enabled agents learned via shrinking boundaries,
- A robust process for skill repertoire expansion through the transition mechanism, and
- A versatile meta-controller capable of performing various skills that can be incrementally expanded at deployment.

II. Related Works

Traditional Controllers. For a long time, roboticists have tried to emulate the efficiency and elegance of locomotions in nature [12], [13]. A gait is a periodic sequence of foot contacts with the ground that produces locomotion [14] and is achievable via analytical models of the dynamics of a legged system [15] or by using central pattern generators (CPG) to produce oscillatory motion patterns and using sensors for close-loop trajectory optimization [16]–[18]. Model predictive control (MPC) is a popular type of approach that produces optimized and distinct gaits [19]. These have been extensively used for quadruped robots, with simplified dynamics and convex optimization [20], linearizing the dynamics and formulating the problem with quadratic programming [21], as well as approaches that use the full dynamics of the system including feet contacts [22].

Learning-Based Controllers. With deep reinforcement learning (RL), it is possible to train controllers for task-based objectives [23], [24]. Learning-based controllers have been used to tackle challenging terrain using just proprioception [25], as well as with other sensors such as with depth [26] and with LiDAR [27]. In our work, we learn locomotion policies based on a motion imitation framework [28], where the agent imitates a reference animation while interacting with physics-enabled environments [29]–[31]. The controllers can be deployed to the real world using methods such as domain randomization [32] during training or domain adaptation with real-world trajectories [33].

Transition between Controllers. Solving complex tasks requires versatile controllers. A common way to coordinate skills is through a hierarchical architecture of low-level and high-level controllers [3], [34]. A task-based objective, in the form of a reward function, drives the behavior of the controllers [35]–[37]. It is also possible to integrate kinematic motion controllers [2], [38], [39] with other high-level controllers trained to interact with other agents in a physics environment [40], [41].

To avoid costly re-training and better scalability, we want to compose the controllers sequentially rather than mix them hierarchically. Sequential execution of learned skills is not a recent idea [42], [43]. Also called skill chaining [9], [44], it has been used to tackle long-horizon tasks [10], splitting a complex task into separate sub policies and coordinating their execution [45], [46]. Other approaches include discovering transition timing via statistical methods conditioned on motion phases [4], using neural nets conditioned on latent states [47], and parametrized transitions [48], [49].

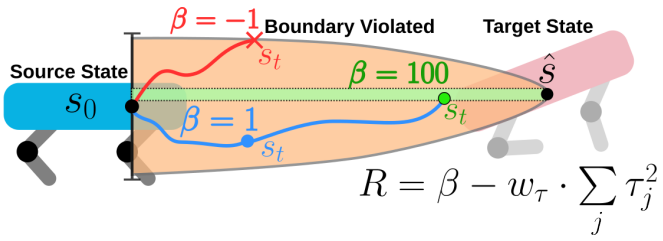

Fig. 2. Illustration of physical-property bounds on the agent's states.

Most similar to ours is the work of [8], where each expert has a matching transition policy that drives the agent to a feasible starting state. The policies are trained by rolling out transition episodes, labeling trajectories a success or failure, and optimizing with RL over the expected success rate. In our proposed work, the composer policy is trained completely in isolation from the other experts. Rather than explicitly optimizing for the success rate, it is trained to drive the robot to a target state. Furthermore, we train only a single policy to transition between any pair of experts.

III. Method

We begin by training independent controllers. These single-skill experts are physics-based controllers trained via motion imitation [28], [33]. Then, we introduce the composer policy $\mathbf{P}$, a module that generates trajectories to arbitrary agent states, allowing the agent to transition between learned skills freely. $\mathbf{P}$ learns to drive the agent from a starting state to a target state sampled from the distribution of the target skill. Expansion is done by simply adding a new expert and using $\mathbf{P}$ to mediate transitions with the existing experts.

A. Experts Initialization

Each expert is a reinforcement learning policy trained by tracking a reference motion clip in a physics simulator with a motion imitation framework. Suppose we have k number of skills, each described by a reference motion clip. Each expert is denoted as a policy $\pi_i(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{g}_i);\ i \in 1...k$ that outputs the actions $\boldsymbol{a}$, executed with a PD controller. The actions are conditioned on the current state of the character $\boldsymbol{s}_t$ as well as data from the reference motion clip $\boldsymbol{g}_i$ from four future frames [33]. Each expert policy trains on a single reference motion clip. Our reward function design is similar to existing motion imitation frameworks [28].

We employ domain randomization extensively to ensure each expert is robust enough to be deployed with the real-world robot. Specifically, we randomize the mass of each link in the robot, introduce disturbance forces, add noise to sensor readings, and randomize the terrain height and friction. The details of the training configuration for each expert are further discussed in Section IV.

B. Composer Policy $\mathbf{P}$ for Novel Transitions

The composer policy $\mathbf{P}$ is an intermediate policy that drives the agent to a target state by generating novel transition trajectories. $\mathbf{P}(\boldsymbol{a}_t|\boldsymbol{s}_t, \hat{\boldsymbol{s}})$ takes in the current state of the

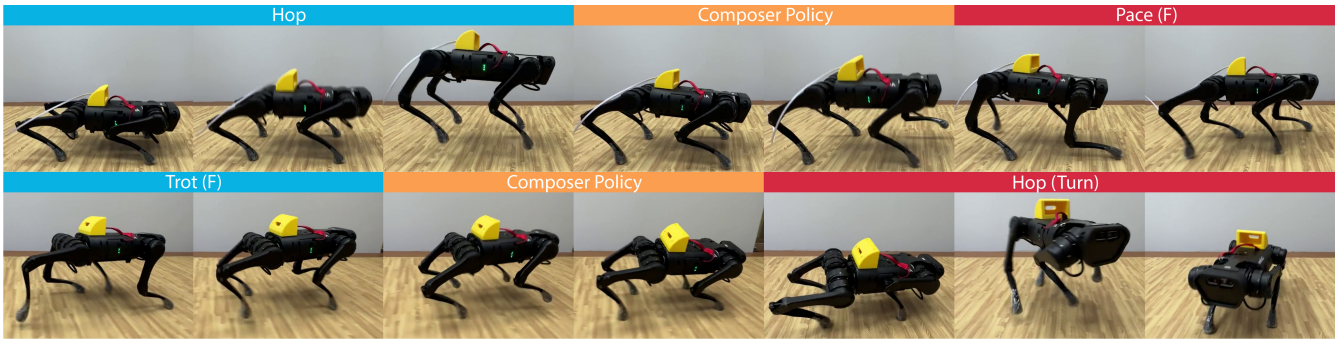

Fig. 3. A visualization of the composer policy with the real-world robot. Please watch the supplementary video for more extensive demonstrations.

character $\boldsymbol{s}_t$, and a target state $\hat{\boldsymbol{s}}$. The state $\boldsymbol{s}_t$ is the same as the experts and consists of position, orientation, binary feet contact indicators, linear and angular velocities, and joint angles and velocities. The fixed target state $\hat{\boldsymbol{s}}$ contains the same information as $\boldsymbol{s}_t$, minus the feet indicators, and represents the state of the agent should be after the transition.

$\mathbf{P}$ is trained with episodic reinforcement learning in a completely independent process from the other experts. During training, we sample the source and target states (with added randomizations) directly from the animation data. During each episode, the goal of $\mathbf{P}$ is to match the target state $\hat{\boldsymbol{s}}$. Potential-based rewards [50] could be used to incentivize the agent to match the properties of the target state. However, in early experiments, we found it difficult to tune the reward elements to ensure the agent matched *all* properties of the target state. Instead, we train $\mathbf{P}$ with a simple indicator function based on a set of narrowing hard boundaries with only a single penalization term for energy efficiency.

The reward function incentivizes the agent to stay within a set of boundaries for each relevant property: joint positions, linear and angular velocity, and orientation [11]. It establishes tolerance regions where the agent can remain and receive a reward of 1. When the agent violates a boundary, the punishment is a negative reward and termination of the episode. If the agent remains within the tolerances until the end of the episode, it receives a large positive reward. Critically, the boundaries are annealed around a line that connects the initial transition state to the target state. The agent learns to stay clear of the shrinking boundaries while closing into the target state (Figure 2). The transition duration is dynamic, and an episode finishes early if all states match the final tolerances. In fact, we expect and observe that most transitions do terminate early if $\mathbf{P}$ is working well. The boundary indicator function has the form

$$\beta = \begin{cases} 100, & \text{if } \max(|\boldsymbol{s}_t - \hat{\boldsymbol{s}}| - \boldsymbol{\sigma}_e) \leq 0 \\ -1, & \text{if } \max(|\boldsymbol{s}_t - \Psi(\boldsymbol{s}_0, \hat{\boldsymbol{s}}, t)| - \boldsymbol{\sigma}_t) \geq 0 \\ 1, & \text{otherwise} \end{cases}, \quad (1)$$

where $\Psi(\boldsymbol{s}_0, \hat{\boldsymbol{s}}, t)$ is a linear function that connects the initial state of the agent $\boldsymbol{s}_0$ to the target state $\hat{\boldsymbol{s}}$, and the tolerance $\boldsymbol{\sigma}_t$ is annealed at each time step t according to the equation

$$\boldsymbol{\sigma}_t = \boldsymbol{\sigma}_s + t^p \ (\boldsymbol{\sigma}_e - \boldsymbol{\sigma}_s), \quad (2)$$

with $\boldsymbol{\sigma}_s$ and $\boldsymbol{\sigma}_e$ denoting the tolerance at the start and at the end of the transition period, and p is an exponential parameter to modify how the boundary shrinks (Table II). With boundary annealing, the composer policy learns to match the target state closely. However, it does not guarantee that the trajectories are smooth or energy efficient. For this, we introduce a penalization term for joint torques. The complete reward function is as follows, where τ_j is the torque of j-th joint of the agent and w_τ is a scalar that controls the scale of penalization,

$$R = \beta - w_\tau \sum_j \tau_j^2. \quad (3)$$

C. Composition of Experts

After training the composer policy $\mathbf{P}$, we can sequentially compose the experts (Figure 4). The agent may start executing a pace skill with π_{pace}. At some later point, an event triggers a switch to a different expert, such as π_{hop}. To execute this transition, $\mathbf{P}$ takes control of the agent and samples a target state $\hat{\mathbf{s}}$ from within the distribution of π_{hop}, using its animation data. Then, $\mathbf{P}$ performs actions such that the character's state at the end of the transition is close enough to the target state that the new expert can take over control. The transition ends with the target policy π_{hop} taking over control of the character. This process can be repeated indefinitely and robustly for any pair in the policy library.

If $\mathbf{P}$ is trained with a large enough distribution of start and target states, it is reasonable to assume that it should learn feasible transitions between any experts, even new ones. This

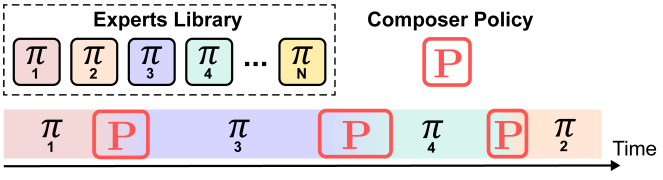

Fig. 4. Illustration of the composer policy $\mathbf{P}$ enables the sequencing of any expert policy from the library over time.

is possible because $\mathbf{P}$ only requires a target state and does not directly depend on or use information from the experts in the library. Therefore, the expansion can be done incrementally and indefinitely by adding new experts and using the same $\mathbf{P}$, without re-training or fine-tuning.

IV. Implementation Details

A. Hardware and Locomotion Experts

Our agent is a Unitree A1 quadruped robot with 12 actuators. We implement a motion imitation framework [28], [33] with Isaac Gym [51] and use a consumer-grade laptop equipped with an Intel 8-core i7-11800H 2.3 GHz and an NVIDIA RTX 3070 8GB with the PPO-clip loss [52] to train the experts. To transfer the policies from simulation to the real world, we employ domain randomization [53], [54] with parameters shown in Table I. The agent states include linear velocity estimated in the real world by fusing IMU readings and the leg velocity during feet contacts. The locomotion experts are deployed in a zero-shot manner and can execute for long periods without failures.

B. Details of the Composer Policy

The architecture of the composer policy $\mathbf{P}(\boldsymbol{a}_t|\boldsymbol{s}_t, \hat{\boldsymbol{s}})$ is a 2-layer feed-forward neural network with 512 and 256 hidden units. Each layer uses ELU activations except for the linear output layer. The composer policy receives an observation vector of 208 dimensions which consists of the three latest states of the agent, the actions from the past three timesteps, the target state $\hat{\boldsymbol{s}_t}$, and the centers of the tolerance boundaries at the current timestep computed from $\Psi(\boldsymbol{s}_0, \hat{\boldsymbol{s}}, t)$. The observations also include a single scalar that encodes the normalized time, starting at 0.0 and increasing to 1.0 at the end of the transition period (episode). The state $\boldsymbol{s}_t$ consists of position, orientation, binary feet contact indicators, linear and angular velocities, and joint angles and velocities. The target state $\hat{\boldsymbol{s}}$ is the same, minus the feet indicators. The maximum episode length is set to 2 seconds. The composer policy outputs 12 target joint angles, actuated using a PD-controller. Table II contains detailed parameters for the tolerance boundaries and reward penalization terms.

V. Experiments and Results

We constructed a library of 9 distinct experts in the following motions: *Trot (F)*, *Trot (B)*, *Pace (F)*, *Pace (B)*, *Hop*, *Hop (Turn)*, *Spin*, *Side Step (F)ast*, and *Side Step (S)low*. This results in a total of 72 unique transition pairs. To evaluate generalization to new skills, $\mathbf{P}$ was trained by sampling starting and target states from just 4 experts: *Trot (F)* and *(B)*, *Pace (F)* and *Side Step (F)*. The 5 remaining experts were added in a supposed expansion process, i.e. were not part of the training. We evaluated the composer policy with two target sampling methods. **Composer Policy (R)** samples the target from a random phase of the animation of the target expert. **Composer Policy (O)** samples from an optimal phase interval after analysis of the success rate of **(R)**, detailed in Section V-C.We compare our approach to two naive baselines and three existing methods from the literature:

TABLE I

Parameters for domain randomization. Ranges are sampled uniformly. The feet contacts value indicates the probability of zeroing out the contacts, per foot.

Parameter	Value	
	Experts	**Composer Policy**
Action Noise	± 0.02	± 0.02
Rigid Bodies Mass	[75%, 125%]	[95%, 105%]
P Gain (PD Controller)	[35, 65]	[45, 55]
D Gain (PD Controller)	[1.0, 1.4]	[0.9, 1.2]
Ground Friction	[0.1, 1.5]	[0.1, 1.5]
Noise - Orientation	± 0.05	± 0.06
Noise - Linear Velocity	± 0.25	± 0.25
Noise - Angular Velocity	± 0.3	± 0.3
Noise - Joint Angles	± 0.02	± 0.02
Noise - Joint Velocity	–	± 1.5
Noise - Feet Contacts	20%	20%

TABLE II

Tunable Parameters of the composer policy. σ_s and σ_e indicate the tolerance at the start and end of the transition, respectively.

Component	Value	σ_s	σ_e
CoM - Height	$p = 2$	0.35	0.02
Orientation	$p = 4$	1	0.2
Linear Velocity	$p = 8$	2.5	0.2
Angular Velocity	$p = 8$	15	0.2
Joint Angles	$p = 2$	3.14	0.5
Torque Term	$w_\tau = 0.0001$	–	–

- **Random Switch** – Instantaneously switches control from the source to the target expert at a random time.
- **Linear Interpolation** – Linearly interpolates the target joint angles from the start to the target state for 0.5 seconds.
- **Mixture-of-Experts (MoE)** [1] – A gating network is trained to output a 9-dimensional vector of activation weights for each expert. **MoE** was trained with access to all 9 experts. A new target expert was randomly selected every 3 seconds. The policy was rewarded for matching the animation of the selected expert.
- **Transition Motion Tensor (TMT)** [4] – The TMT performs an instantaneous switch to the target expert at an optimal timing. The timing is determined offline by a Monte Carlo approach to determine the source and target phases with the highest success rate.
- **Target State Proximity (TSP)** [8] – A policy with a reward function in the style proposed by [8]: $R = P(s_{t+1}) - P(s_t)$. Originally, the proximity function $P(s)$ is modeled by a neural-net, but, because we have access to the animation of the target expert, we sample it directly and compute the euclidean distance. The **TSP** is trained and evaluated under the same settings as our **Composer Policy**. However, **TSP** requires one transition policy per expert. Based on motion similarity, the model *Trot (B)* was used for the expansion skill *Pace (B)*, *Side Step (F)* for *Side Step (S)*, and *Pace (F)* for the remaining expansion skills.

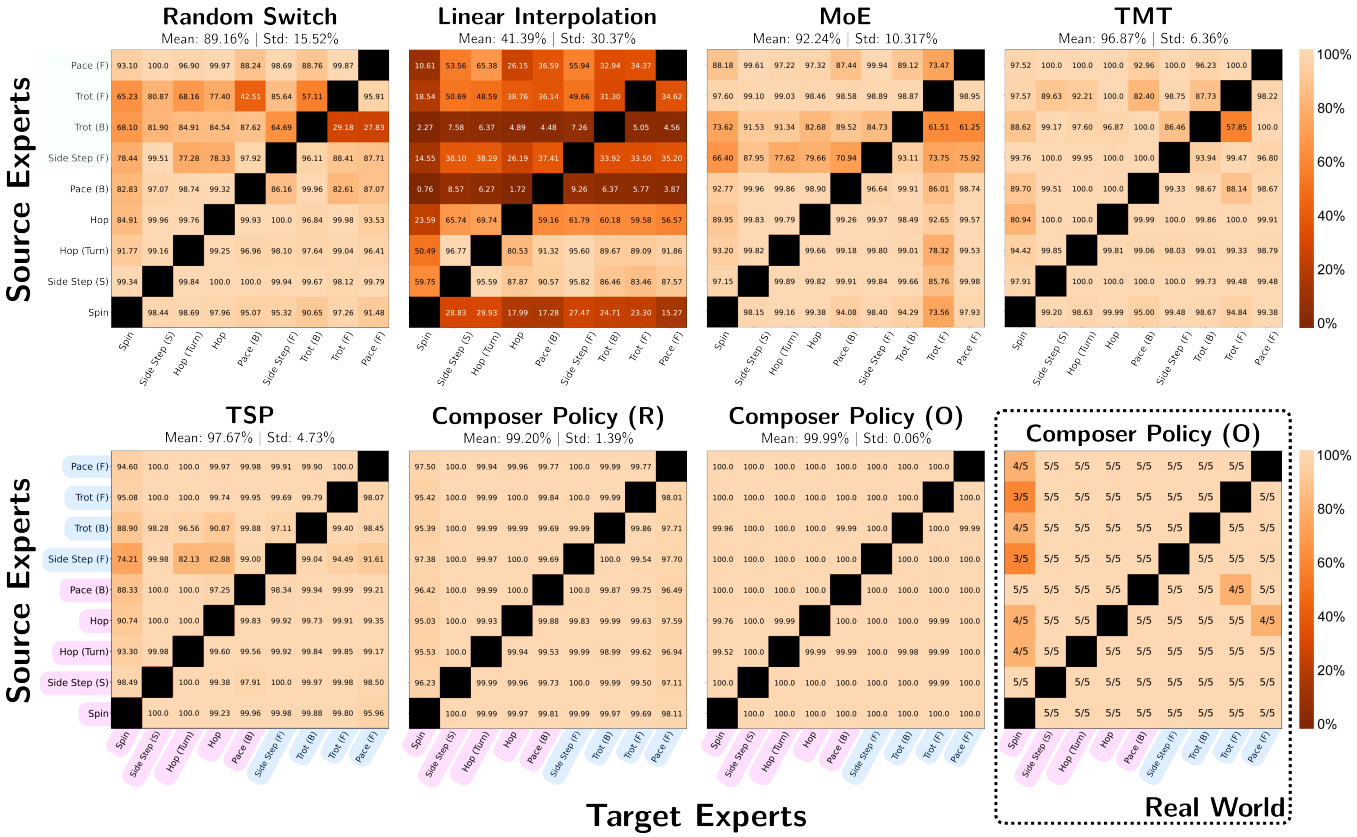

Fig. 5. Success rates of the composition strategies for 72 transition pairs. (F) and (B) indicate forward and backward for Trot and Pace, (F)ast and (S)low for Side Step. Experts highlighted in blue were part of the training set (N=4), and purple indicates new experts added to the library after training (N=5). Our **Composer Policy (R)** and **(O)** variants outperforms the baselines and existing approaches in simulation by a significant margin. It is also successful in the real-world, with just 10 failures out of 360 trials.

A. Success Rate in Simulation

For each strategy, an evaluation episode consisted of three stages: pre-transition, transition, and post-transition. In the pre-transition stage, the source expert is executed. It is followed by the transition stage where one of the methods takes control of the agent. Finally, the target expert takes control for 10 seconds in the post-transition stage. A transition is considered a failure if any links other than the feet of the agent touch the ground during the transition or post-transition stages. For each pair and method we executed 50k evaluation episodes, the results of which are consolidated in Figure 5.

The baseline **Random Switch** provides some information on the difficulty of the transition pairs. We can observe that the lowest success occurred for pairs with distinct motions, e.g. moving from forward to backward, and sideways. The **Linear Interpolation** baseline performed the worst with an average success under 50%. **MoE** beat the baselines in challenging motions, but the low success rate combined with degradation of motion quality is undesirable. **TMT** uses a simple transition mechanism and managed to achieve a high success rate, but performed poorly for a few pairs. **TSP** was the second-best performing method, with poor performance in few pairs, but specially when the target expert was the out-of-distribution *Spin* motion. Our approach **Composer Policy** outperformed the other comparison methods. The random sampling mode **(R)**, shows that the composer policy can accommodate well the complete range of source and target phases. The performance can be further improved by picking only the best target phases, as demonstrated by the almost perfect success rate of the **(O)**ptimal sampling method.

B. Success Rate of the Composer Policy in the Real-World

We deployed the same library of experts and composer policy used in the simulation setting to a real-world A1 quadruped robot. For each pair available in the library we executed 5 trials. With 72 available pairs, this resulted in total of 360 trials. A real-world trial consisted of the same three stages of the simulation. However, due to limitations of our physical space, the post-transition stage was limited to 4 seconds. The transition stage was triggered randomly between 2 to 3 seconds after the pre-transition stage. The real-world results are also shown in Figure 5 (bottom right). We can observe that the simulation performance translated well to the real world, with only 10 failures out of 360 trials, resulting in an average success rate of 97.22%. Figure 3 shows deployment of a transitions that starts airborne and into a forward moving motion.

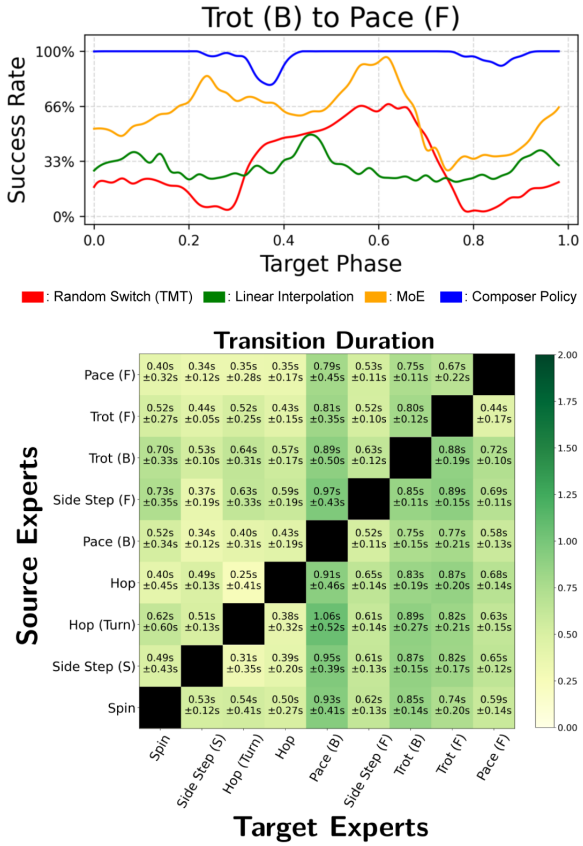

Fig. 6. Success rate across all target phases (top) and transition duration of our composer policy (bottom). By excluding regions of lower success we sample targets from an optimal interval.

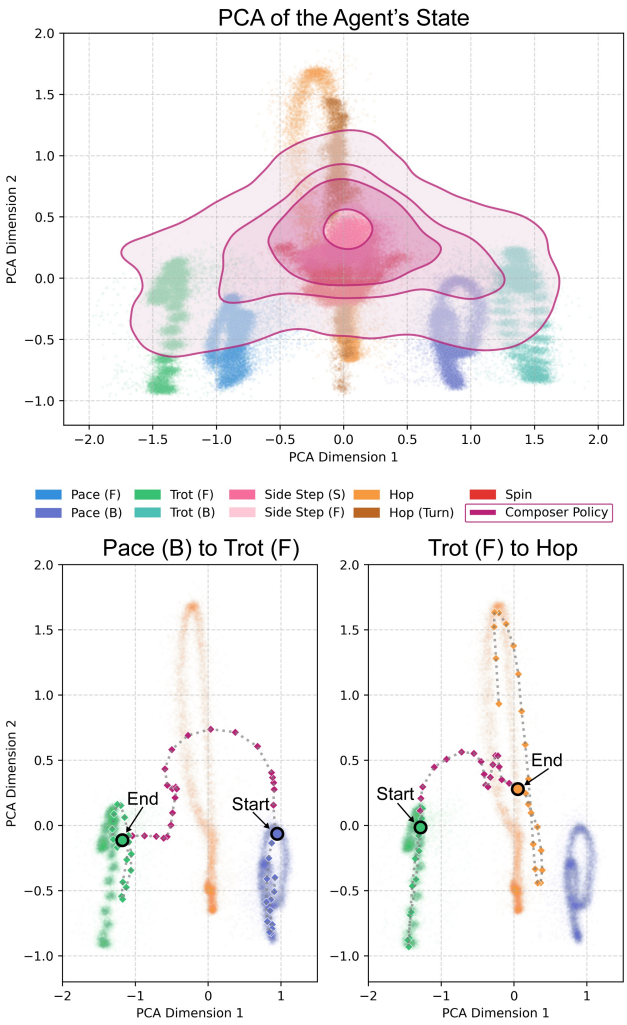

Fig. 7. PCA visualization of the experts distribution (top) and transition trajectories (bottom). The purple contours depicts the density distribution of the Composer Policy's states, where the outermost includes 95% of the data. The bottom trajectories show that our method can drive the agent's state between the different distributions effectively.

C. Transition Analysis

We measure the transition success rates for a difficult motion pair across target phases in Figure 6 (top), showing the performance of the **Composer Policy (R)** for every target phase. By excluding intervals where the success rate is lower than desired, i.e. less than 90% the **(O)**ptimal sampling method is obtained. Figure 6 (bottom) shows that transitions between experts moving in the same direction are shorter than for opposing motions. Note that none approaches the 2-second limit, which means the composer policy can guide the agent effectively toward the target state for all tested pairs. Next, we demonstrate the robust generalization capability of the composer policy with a PCA plot of the agent's state in Figure 7 (top). The outermost contour contains 95% of the samples. We can observe that the distribution of the composer policy is stretched to overlap the majority of the experts' distribution. As long as there is an overlap between the experts and composer policy distribution we should be able to execute a transition. We also provide two examples of transition trajectories, demonstrating the composer policy's capacity to take the state of the agent from one expert to another (Figure 7 bottom).

VI. Discussion

We present an approach that allows incremental expansion of a skill repertoire without re-training. Rather than hierarchically mixing the experts, we train a composer policy to transition between any pairs of experts. Our method outperforms existing approaches in simulation, achieves a high success rate in the real world, and generates smooth transitions with short duration.

Our method assumes that the experts are periodic and have an easily accessible distribution of target states that we can sample from. We show that it works well for a large distribution, however, it might not apply to all types of policies, such as perception-based controllers. Overcoming these limitations are critical for building a diverse and expansible skill repertoire for any type of skills.

References

[1] J. Won, D. Gopinath, and J. Hodgins, "A scalable approach to control diverse behaviors for physically simulated characters," *ACM*

Trans. Graph., vol. 39, no. 4, Jul. 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392381

[2] 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.

[3] X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine, "Mcp: Learning composable hierarchical control with multiplicative compositional policies," in *NeurIPS*, 2019.

[4] J. H. Soeseno, Y.-S. Luo, T. P.-C. Chen, and W.-C. Chen, "Transition motion tensor: A data-driven approach for versatile and controllable agents in physically simulated environments," in *SIGGRAPH Asia 2021 Technical Communications*, 2021, pp. 1–4.

[5] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Trans. Graph.*, vol. 21, no. 3, p. 473482, Jul. 2002. [Online]. Available: https://doi.org/10.1145/566654.566605

[6] A. Safonova and J. K. Hodgins, "Construction and optimal search of interpolated motion graphs," *ACM Trans. Graph.*, vol. 26, no. 3, p. 106es, Jul. 2007. [Online]. Available: https://doi.org/10.1145/1276377.1276510

[7] L. Liu, M. V. D. Panne, and K. Yin, "Guided learning of control graphs for physics-based characters," *ACM Trans. Graph.*, vol. 35, no. 3, may 2016.

[8] Y. Lee, S.-H. Sun, S. Somasundaram, E. S. Hu, and J. J. Lim, "Composing complex skills by learning transition policies," in *International Conference on Learning Representations*, 2018.

[9] A. Bagaria and G. Konidaris, "Option discovery using deep skill chaining," in *International Conference on Learning Representations*, 2019.

[10] Y. Lee, J. J. Lim, A. Anandkumar, and Y. Zhu, "Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization," *Proceedings of Machine Learning Research*, vol. 164, pp. 406–416, 2021.

[11] L.-K. Ma, Z. Yang, T. Xin, B. Guo, and K. Yin, "Learning and exploring motor skills with spacetime bounds," *Computer Graphics Forum*, vol. 40, no. 2, 2021.

[12] R. M. Alexander, "The gaits of bipedal and quadrupedal animals," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 49–59, 1984.

[13] D. F. Hoyt and C. R. Taylor, "Gait and the energetics of locomotion in horses," *Nature*, vol. 292, no. 5820, pp. 239–240, 1981.

[14] G. C. Haynes and A. A. Rizzi, "Gaits and gait transitions for legged robots," in *Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2006, pp. 1117–1122.

[15] P.-B. Wieber, R. Tedrake, and S. Kuindersma, "Modeling and control of legged robots," in *Springer handbook of robotics*. Springer, 2016, pp. 1203–1234.

[16] Y. Fukuoka, Y. Habu, and T. Fukui, "Analysis of the gait generation principle by a simulated quadruped model with a cpg incorporating vestibular modulation," *Biological cybernetics*, vol. 107, no. 6, pp. 695–710, 2013.

[17] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural networks*, vol. 21, no. 4, pp. 642–653, 2008.

[18] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2554–2561.

[19] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 577–584.

[20] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1–9.

[21] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8484–8490.

[22] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[23] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[24] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Conference on Robot Learning*. PMLR, 2023, pp. 22–31.

[25] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[26] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on Robot Learning*. PMLR, 2023, pp. 403–415.

[27] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[28] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018.

[29] L. Mourot, L. Hoyet, F. Le Clerc, F. Schnitzler, and P. Hellier, "A survey on deep learning for skeletonbased human animation," *Comput. Graph. Forum*, vol. 41, no. 1, pp. 122–157, Feb. 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/cgf.14426

[30] C. Li, S. Blaes, P. Kolev, M. Vlastelica, J. Frey, and G. Martius, "Versatile skill control via self-supervised adversarial imitation of unlabeled mixed motions," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2944–2950.

[31] E. Vollenweider, M. Bjelonic, V. Klemm, N. Rudin, J. Lee, and M. Hutter, "Advanced skills through multiple adversarial motion priors in reinforcement learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5120–5126.

[32] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[33] 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*, 07 2020.

[34] J. Won, D. Gopinath, and J. Hodgins, "A scalable approach to control diverse behaviors for physically simulated characters," *ACM Trans. Graph.*, vol. 39, no. 4, aug 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392381

[35] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Trans. Graph.*, vol. 40, no. 4, 2021.

[36] D. Jain, A. Iscen, and K. Caluwaerts, "Hierarchical reinforcement learning for quadruped locomotion," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7551–7557.

[37] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and efficient locomotion via learned gait transitions," in *5th Annual Conference on Robot Learning*, 2021.

[38] S. Starke, H. Zhang, T. Komura, and J. Saito, "Neural state machine for character-scene interactions," *ACM Trans. Graph.*, vol. 38, no. 6, Nov. 2019.

[39] S. Starke, Y. Zhao, T. Komura, and K. Zaman, "Local motion phases for learning multi-contact character movements," *ACM Trans. Graph.*, vol. 39, no. 4, Jul. 2020.

[40] K. Bergamin, S. Clavet, D. Holden, and J. R. Forbes, "Drecon: Data-driven responsive control of physics-based characters," *ACM Trans. Graph.*, vol. 38, no. 6, p. 206, 2019.

[41] S. Park, H. Ryu, S. Lee, S. Lee, and J. Lee, "Learning predict-and-simulate policies from unorganized human motion data," *ACM Trans. Graph.*, vol. 38, no. 6, Nov. 2019.

[42] J. Kober, J. Peters, J. Kober, and J. Peters, "Movement templates for learning of hitting and batting," *Learning Motor Skills: From Algorithms to Robot Experiments*, pp. 69–82, 2014.

[43] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 763–768.

[44] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," *Advances in neural information processing systems*, vol. 22, 2009.

[45] Y. Lee, J. Yang, and J. J. Lim, "Learning to coordinate manipulation skills via skill behavior diversification," in *International conference on learning representations*, 2019.

[46] A. Clegg, W. Yu, J. Tan, C. K. Liu, and G. Turk, "Learning to dress: Synthesizing human dressing motion via deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–10, 2018.
[47] G. Christmann, Y.-S. Luo, J. H. Soeseno, and W.-C. Chen, "Expanding versatility of agile locomotion through policy transitions using latent state representation," *arXiv preprint arXiv:2306.08224*, 2023.
[48] G. C. Haynes, F. R. Cohen, and D. E. Koditschek, "Gait transitions for quasi-static hexapedal locomotion on level ground," in *Robotics Research*. Springer, 2011, pp. 105–121.
[49] C. Boussema, M. J. Powell, G. Bledt, A. J. Ijspeert, P. M. Wensing, and S. Kim, "Online gait transitions and disturbance recovery for legged robots via the feasible impulse set," *IEEE Robotics and automation letters*, vol. 4, no. 2, pp. 1611–1618, 2019.
[50] B. Badnava, M. Esmaeili, N. Mozayani, and P. Zarkesh-Ha, "A new potential-based reward shaping for reinforcement learning agent," in *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2023, pp. 01–06.
[51] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.
[52] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[53] G. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid Locomotion via Reinforcement Learning," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.
[54] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," in *Robotics: Science and Systems*, 2021.