2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Expanding Versatility of Agile Locomotion through Policy Transitions Using Latent State Representation

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

***Abstract*— This paper proposes the *transition-net*, a robust transition strategy that expands the versatility of robot locomotion in the real-world setting. To this end, we start by distributing the complexity of different gaits into dedicated locomotion policies applicable to real-world robots. Next, we expand the versatility of the robot by unifying the policies with robust transitions into a single coherent meta-controller by examining the latent state representations. Our approach enables the robot to iteratively expand its skill repertoire and robustly transition between any policy pair in a library. In our framework, adding new skills does not introduce any process that alters the previously learned skills. Moreover, training of a locomotion policy takes less than an hour with a single consumer GPU. Our approach is effective in the real-world and achieves a 19% higher average success rate for the most challenging transition pairs in our experiments compared to existing approaches.**

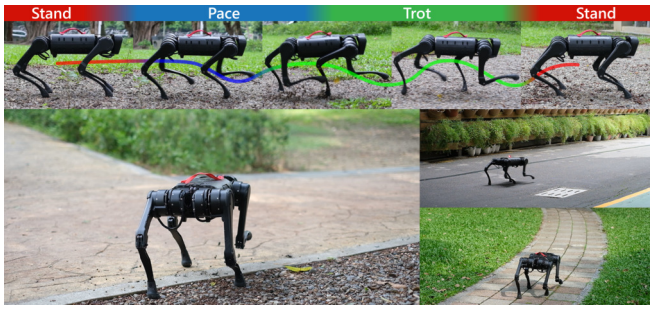

Fig. 1. Our meta-controller enables the transition between locomotion policies in various real-world environments. We adopt three representative gaits within the robot's capabilities to perform static-to-dynamic, dynamic-to-dynamic, and dynamic-to-static transitions.

I. Introduction

Robotics is impactful in many applications, from friendly human companions to industrial robots for automation and specialized robots in exploration industries. All these applications require robust and versatile movements to complete the assigned tasks. For instance, when exploring challenging terrains, the robot needs to handle unexpected disturbances and be versatile to maneuver through obstacles. These properties are enabled by locomotion policies that coherently choreograph every joint of the robot. However, as the skillset of the robot expands, it becomes exponentially intricate to maintain movement robustness.

To tackle this problem, it is common to develop an initial locomotion policy in simulation using deep reinforcement learning. This initial policy can be optimized for various objectives such as energy consumption [1], task-based objectives [2], and imitating motions from a reference animation clip [3]. Next, these policies are transferred to the real-world setting through domain randomization [4], and domain adaptation [5] methods. Although this setup can produce agile locomotion policies in the real-world, they struggle with expanding the skill repertoire of the robot. It is because the setup requires the re-training or fine-tuning of the initial policy in simulation and repeating the same transfer process. This modification further causes intricacies as the added skills impose varying complexity and require the delicate design of reward functions.

We adopt an alternative setup, where the complexity of each skill is contained into independent policies [6], with flexibility of the architecture and objective function. The skill repertoire is represented by a library of policies which can include high-level directives [7], [8]. We start by training robust policies that can be transferred to a real quadruped. Next, to expand the versatility of the robot, we introduce a transition mechanism between the policies. As pointed out by [3], [6], [9]–[12], it is possible to use the phase variable as a low-dimensional proxy for the state of the agent. While this approximation can be successful in simulation, it is less accurate in a real-world setting since the same phase value can represent two significantly different states of the agent.

In our work, we use the latent state representations of the policies to encode the state of the agent and its environment conditions. We design a robust learning-based transition strategy that identifies successful transitions across all policy pairs. We then compose a meta-controller that regulates transitions between policies acting on the robot (see Figure 1). We summarize our contributions as follows,

- A robust transition strategy, *transition-net* that can accurately identify viable transitions between policies in the real-world using latent state representations;
- A scheme to iteratively expand the policy library of the robot without altering existing policies; and
- A demonstration and evaluation of the proposed method with a real-world quadruped robot.

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

II. Related Works

Non-Learning Controllers. Legged locomotion for robotics is a field with a large body of work that aims to emulate the efficiency and agility of the locomotion gaits observed in nature [13], [14]. A legged robot is able to move by producing a periodic sequence of foot contacts with the ground, called a gait [15]. For a long time, gaits have been produced via analytical models of the dynamics of a legged system [16]. Traditional approaches include using central pattern generators (CPG) to produce an oscillatory motion pattern that is further optimized into a desired trajectory [17]–[19]. Model predictive control (MPC) is another approach used to generate optimized trajectories resulting in various gaits [20]–[23]. While many works focus on optimizing a single gait, others also made efforts toward optimizing multiple gaits and enabling transitions [24]–[26].

Learning-Based Controllers. Recently, deep reinforcement learning has become a popular approach to tackle challenging and agile locomotion [27]. It doesn't require an accurate model of the system dynamics to achieve robust locomotion [5], [28], [29]. To enable the emergence of agile locomotion gaits, it is necessary to carefully design a reward function [29] that incentivizes the intended behavior and punishes undesired motions. This process [30] is laborious and time-consuming, and the resulting reward parameters can be brittle. Incorporating motion priors, such as modulating trajectory generators [31], can improve the overall robustness and learning stability [28], [32]. Another approach that produces natural-looking gaits is learning from reference motions [3], [33]. Using information from motion capture data from real animals, the agent is conditioned to simultaneously imitate a reference clip and execute a goal-oriented task [3], [8], [34], [35].

Simulation-to-Real Policy Transfer. To ensure policies can be transferred to the real world (sim-to-real), domain randomization (DR) [4] can be applied during training in simulation. By randomizing the physics parameters and the agent's properties during training, the policy becomes robust in a variety of environments [36]. Performance can be further improved by adapting the policies via system identification in the real-world setting [5], [32], [37]. A common approach is to encode the physics parameters into a latent representation [5], [28], [29]. Then, trajectories sampled from the real-world can be used to search the latent space at deployment time [5], or estimated from the recent history of states and actions [28], [29]. In the context of policy transitions, previous work has shown great success in simulation [6], but suffer from limitations in the real-world that we aim to address.

In our work, we develop locomotion policies that learn different gaits through motion imitation and can be deployed in a real-world setting via domain randomization. Distinctly from previous works, each gait of our library is learned as an independent locomotion policy. To expand the versatility of the overall locomotion, we employ a meta-controller that robustly unifies these independent policies through the use of our novel transition mechanism.

III. Methodology

Our goal is to enable real-world quadruped robots to incrementally expand their library of locomotion gaits without altering previously learned ones. For this purpose, we contain the complexity of the gaits by training independent policies that specialize in a particular gait. Then, we construct a library of robust policies that can transfer to real-world robots with the use of domain randomization. Similar to [6], we introduce a transition mechanism to link the independent policies by instantaneously switching between any two arbitrary policies. But, where previous works used phase to identify configurations that yield successful outcomes, instead, we propose a transition strategy that uses the latent representations of the policies, the *transition-net*. During deployment with a real-world A1 quadruped robot, we construct a meta-controller that can access all policies available in the library and regulates the switch between an active and a queued policy.

A. Learning Independent Policies for Quadruped Robots

Each policy in our skill repertoire learns a locomotion gait from a reference clip using a motion imitation framework in simulation [5], [36], [38]. Specifically, a policy π is learned by maximizing the expected return,

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)}[\sum_{t=0}^{T} \gamma^t r_t], \tag{1}$$

where $p(\tau|\pi)$ is the likelihood of a trajectory τ given the policy π, and $\sum_{t=0}^{T} \gamma^t r_t$ is the accumulated reward collected during the trajectory. r_t denotes the reward collected at time $t \in T$, where T denotes the length of each episode, and $\gamma \in [0, 1]$ represents the discount factor for future rewards. We train policies similar to [3], [32], where the policy learns an action distribution by imitating a reference motion clip. The input of the policy consists of the agent's state $\boldsymbol{s}_t$ and data from the reference motion clip $\boldsymbol{g}_t$. The policy is modeled as a feed-forward network that outputs the action distribution $\boldsymbol{a}_t$ given the current state and reference motion data $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{g}_t)$. See Section IV for more details.

To transfer to the real-world, we apply extensive domain randomization (DR) of the simulation physics parameters and add other disturbances during the training process. This improves the inherent robustness of each policy and minimizes the performance gap between the simulation and the

TABLE I

Parameters for domain randomization with uniform sampling.

Parameter	Range	Type
Gravity	[0.8, 1.2]	Scaling
Action Noise	[-0.03, 0.03]	Additive
Observation Noise	[-0.03, 0.03]	Additive
Rigid Bodies Mass	[0.85, 1.15]	Scaling
Ground Friction	[0.25, 1.5]	–
Observation Latency	[0.0, 0.020] s	–
Stiffness Gain (PD Controller)	[45, 75]	–
Damping Gain (PD Controller)	[0.9, 1.8]	–

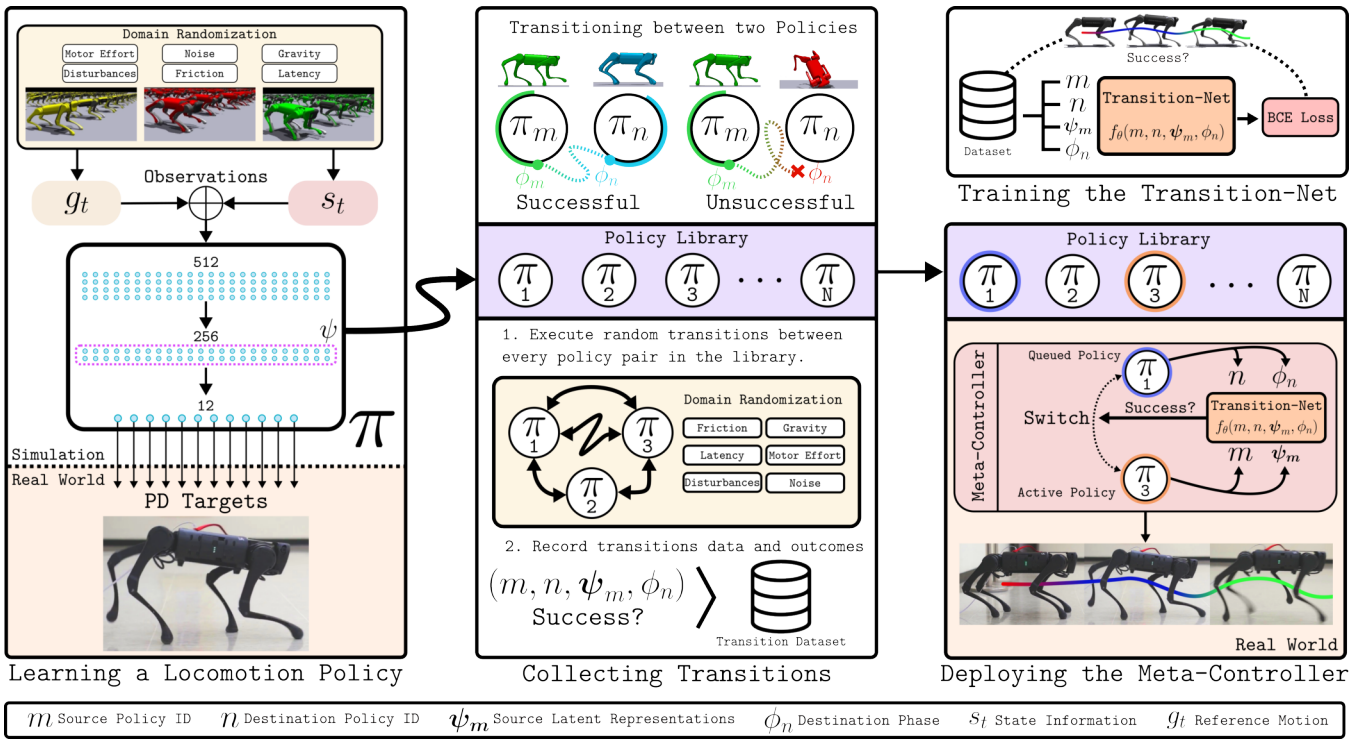

Fig. 2. We start by learning gait policies individually to build a policy library. Then, we collect transitions between every policy pair in the library. Our *transition-net* acts as a robust transition strategy that enables the meta-controller to execute transitions in the real-world.

real-world setting [32], [39], [40]. With careful selection and application of the DR parameters described in Table I, we can reliably perform sim-to-real deployment of the independent locomotion policies with no failures (see Figure 1). For each gait, the training process is realized independently and from scratch. At this point, we have a collection of independent policies that are robust enough to be deployed to real-world robots. However, each policy only enables the robot to perform one specific gait, and a method to transition between them has to be established.

B. Transitioning between Policies with Transition-Net

Each policy is considered a robust periodic controller capable of recovery from unstable states within an unspecified tolerance. For example, when the agent stumbles due to external perturbations, the policy acts to prevent its collapse and resumes its normal periodic motion afterward. Given this behavior, when a policy from the library is executing, it is possible to instantaneously switch the execution to a different policy at a particular destination phase. With proper timing of the switch and a good choice of the destination phase, the new active policy takes control, corrects unstable behavior due to the initial configuration, and resumes its normal periodic execution.

Our method provides a way to consistently identify transition configurations that yield successful outcomes, i.e., where the agent remains stable after a transition. It takes advantage of the rich latent representations of each policy, obtained from the last hidden layer. We formulate a transition function $f_\theta(\cdot)$ that maps the transition configurations to their outcome, where θ represents the weights of a feed-forward neural network, called the *transition-net*. The transition outcome is denoted by the binary variable $\alpha \in \{0, 1\}$, and the transition configuration $\mathcal{C}$ is represented as a tuple of four elements,

$$\mathcal{C} = (m, n, \boldsymbol{\psi}_m, \phi_n), \tag{2}$$

where m and n are identifiers of the source and destination policy, respectively; $\boldsymbol{\psi}_m$ is a high-dimensional vector denoting the latent representations of the source policy (see Figure 2), and $\phi_n \in [0, 1)$ is the phase of the destination policy.

To train the *transition-net*, we collect millions of transition samples in simulation as detailed in Section IV. Using this transition dataset, we train the *transition-net* in a supervised manner solving for a binary classification problem, where it aims to predict whether a transition configuration $\mathcal{C}$ would result in a successful outcome $\alpha = 1$ or a failure $\alpha = 0$. It optimizes for the binary cross-entropy (BCE) loss with α as the classification label y,

$$y log(\hat{y}) + (1 - y) log(1 - \hat{y}), \tag{3}$$

where y denotes the recorded ground truth outcome, and $\hat{y}$ are the network's predictions.

C. Unifying Policies with a Meta-Controller

Next, to coherently unify all policies during deployment, we construct a meta-controller using the *transition-net* estimates as transition scores. The meta-controller queries the

Algorithm 1 Meta-Controller with *Transition-Net* Strategy

Input: Policy library Π and Transition-Net $f_\theta(\cdot)$
$\pi_m \leftarrow$ sample a policy from policy library Π
for each time step $t \in$ Real World **do**
  $(\boldsymbol{a}_t, \boldsymbol{\psi}_m) \sim \pi_m(\boldsymbol{s}_t, \boldsymbol{g}_t)$ ▷ Update step.
  PDTargetControl($\boldsymbol{a}_t$) ▷ Controls the robot.
  $\pi_n \leftarrow$ RequestChangePolicy()
  if $\pi_m \neq \pi_n$ **then**
    $\hat{y} \leftarrow \max(\{f_\theta(m, n, \boldsymbol{\psi}_m, \phi_n); \phi_n \in [0, 1)\})$
    if $\hat{y} > th$ **then**
      $\pi_m \leftarrow \pi_n$ ▷ Switches policy.
    end if
  end if
end for

transition-net to identify the best transition configurations. It is responsible for choosing, scheduling, and executing the policies deployed on the robot using these elements:

- an active policy, which controls the robot by generating joint target angles actuated via PD control;
- a queued policy, to be switched for the active policy as soon as possible; and
- a transition function $f_\theta(\mathcal{C})$ that provides a score given the current configuration $\mathcal{C}$.

During runtime, we start by defining an initial active policy that controls the robot. The active policy can be initialized to any policy available in the library. At some point in time, a request for a policy change happens, and a different policy from the library is queued. Once a policy is queued, the meta-controller recognizes that a switch should happen. At every timestep (30Hz), it queries the transition function $f_\theta(\mathcal{C})$ and computes the transition score of switching the active policy for the queued policy. Note that we search over multiple destination phases ϕ_n when querying and choose the highest-scoring one. When the transition score crosses a predefined threshold, the queued policy becomes active and takes control of the robot. The meta-controller pipeline is presented in Algorithm 1.

IV. Implementation Details

A. Quadruped Robot and Simulation Engine

In our experiments we use the Unitree A1 quadruped robot, which has 12 joints actuated via a PD controller. In the training stage, the simulated agent matches the configurations and properties of the real robot. The observation space of the policy is composed of the state $\boldsymbol{s}_t$ and the reference motion $\boldsymbol{g}_t$. The state $\boldsymbol{s}_t \in \mathbb{R}^{102}$ includes state information from the current and past two timesteps of the agent. A single state is composed of 12 joint angles, the orientation and angular velocities (6), a binary indicator of the contact for each foot (4), and the previous actions of the policy (12). The reference motion $\boldsymbol{g}_t \in \mathbb{R}^{56}$ contains the target poses from the motion capture data at 4 future timesteps, up to one second in the future [32]. It functions as an implicit phase variable by modulating the near future targets of the agent. When searching for the best destination phase of the queued policy, we shift the queued reference motion data in the time axis.

TABLE II
Hyperparameters for training a locomotion policy with PPO.

Parameter	Value
Number of Environments	4096
Sequence Length	24
Sequences per Environment	4
Policy Optimization Iterations	5
PPO Batch Size	12288
Adam Optimizer LR	3 x 10^{-4}
Discount factor γ	0.95
Generalized Advantage Estimation λ	0.95
PPO Clip threshold	0.2
KL threshold	0.008
Entropy coefficient	0.0

The gait policies are trained following the imitation learning objective from [3], with the PPO clip loss parameters described in Table II. We developed the RL training environment using the simulator *Isaac Gym* [41], which can accelerate the training by instancing several parallel environments (4096) in a single physics scene and exposing the simulation states via a *PyTorch*-like API. With our implementation, the training process of a single locomotion policy takes less than one hour of wall clock time on a system equipped with an Intel i7-11800H and an RTX 3070 8GB. The policies can be deployed with the real robot in a zero-shot manner.

B. Training of the Transition-Net

Our proposed *transition-net* is trained in a supervised manner with millions of transition samples collected from simulation with domain randomization as described in Table I. The samples contain paired labels of transition configuration and their corresponding outcome, where the source and destination policies are uniformly sampled from the library. Since the random switching strategy used to collect the samples introduce an imbalanced number of failure and successful cases, we sub-sample transition samples such that the number of success and failure samples are balanced.

The *transition-net* is implemented as a feed-forward network with $128 - 64 - 32$ neurons as the intermediate layers, with dropout ($p = 0.4$) and ReLU activation functions applied after each layer except for the output layer, which uses a sigmoid. The neural network is trained for 100 epochs using a mini-batch of 128 samples, and the AdamW optimizer [42] with a learning rate of $5\text{e}{-4}$. During deployment, we use the output of the *transition-net* as a scoring function, and only perform transitions when the score crosses the threshold $f_\theta(\mathcal{C}) > th$; where $th = 0.95$.

V. Experiments

To evaluate the effectiveness of our approach in identifying transition configurations that map to successful outcomes, we construct a policy library consisting of three locomotion gaits, namely, $pace$, $trot$, and $stand$. By using a static gait ($stand$) and two dynamic gaits ($pace$ and $trot$) we are

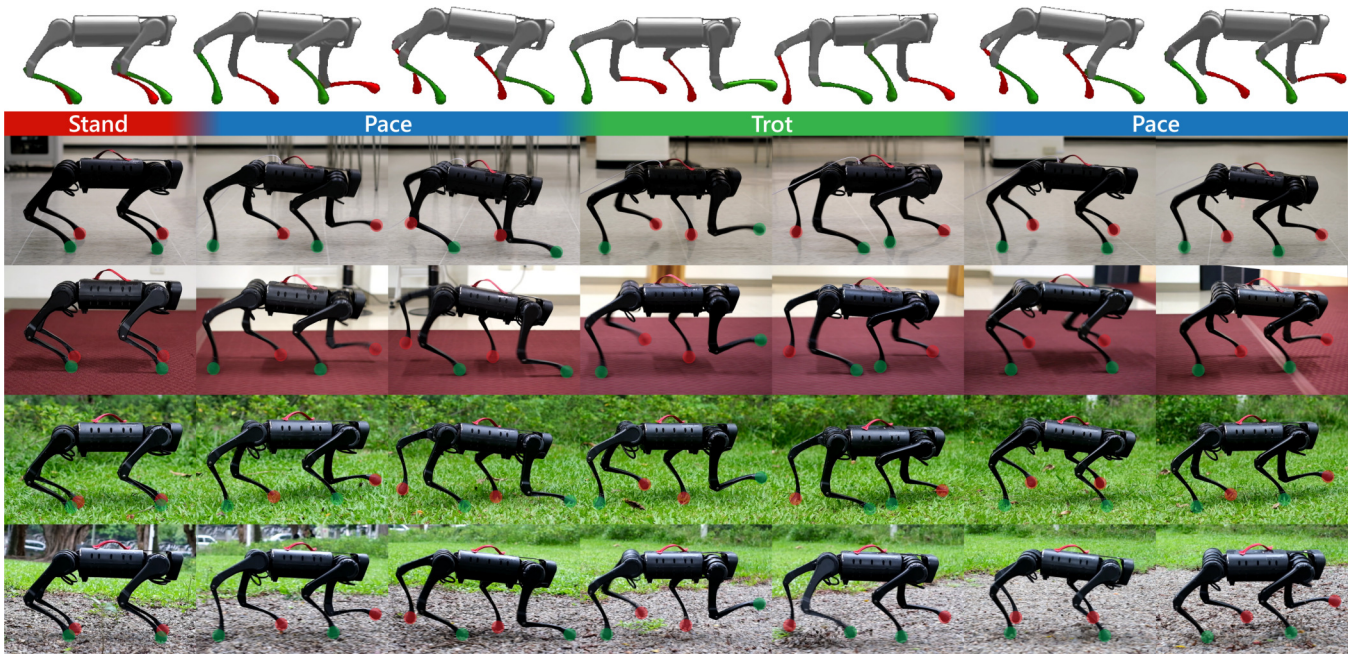

Fig. 3. Transition between locomotion policies in simulation (first row) and in the real-world. Our policies are robust and can be deployed in a variety of environments. For more comprehensive results, please refer to the supplementary video.

able to run experiments that include dynamic-to-dynamic transitions as well as static-to-dynamic and vice-versa. Next, we unify these policies using a meta-controller that uses the *transition-net* to score transition configurations and regulate the timing of a policy switch. Figure 3 provides a visual illustration of the deployed meta-controller in the real-world. The difference between the $pace$ and $trot$ gaits is highlighted by the feet contact pattern, where $trot$ crosses the front and back legs while moving, enabling the robot to move at a faster speed. Notably, the policies can adapt to the different environments shown in the figure without requiring any additional tuning process in the real-world. In addition to the qualitative analysis of the policies, we also evaluate the transition robustness produced by our transition strategy compared to other existing works in Section V-A. Finally, in Section V-B we investigate alternative input representations for the *transition-net*, validating our design choices.

A. Transition Robustness in Simulation and Real-World

To evaluate the robustness of the transitions generated by the ***transition-net (Ours)***, we compare it to four other transition strategies: ***random***, ***value function*** [36], ***transition motion tensor (TMT)*** [6], and ***TMT-value***. ***TMT-value*** is an extension of TMT we developed by integrating value function estimates into the querying function of TMT. We first evaluate the success rate of the transition strategies under increasing disturbance forces in simulation. Each strategy performs every possible transition pair while disturbance forces are applied at random directions and intervals (0.1 to 0.3 seconds). The disturbance magnitude starts at 0 N and increases up to 500 N in increments of 50 N. We collect 5000 transition samples for every combination of transition strategy, transition pair, and increment of disturbance magnitude.

Figure 4 (top) summarizes the success rate of each strategy and transition pair. As expected, randomly switching between policies without any heuristics yields suboptimal and unstable transitions. By employing the timing heuristics from the phase variable, ***TMT*** achieves a success rate of 95% for $pace$–$trot$. While the phase variable is a considerable indicator of when to perform a transition, it still fails to disambiguate between different states with similar phase values, e.g., from external disturbances or other motion inconsistencies. Other strategies such as ***value function***, ***TMT-value***, and ***transition-net*** achieve a higher degree of robustness, highlighted by the higher success rate under the more intense disturbances. This is because more dynamic representations can better capture the state of the robot in relation to its environment dynamics. However, since the policy's value function is not specifically designed to facilitate transitions, ***value function*** and ***TMT-value*** strategies suffer from inconsistencies with lower success rates compared to our approach.

Next, we deploy the meta-controller with the real A1 quadruped robot and once again evaluate the success rate of each transition strategy. We collect 50 transition samples for every possible combination of transition strategy and pair from the library. From Figure 4 (bottom), we can observe that the robustness achieved in the simulation translates fairly well to the real-world setting, albeit with a lower overall success rate. The ***value function*** strategy produces inconsistent transitions. This is expected as the value function

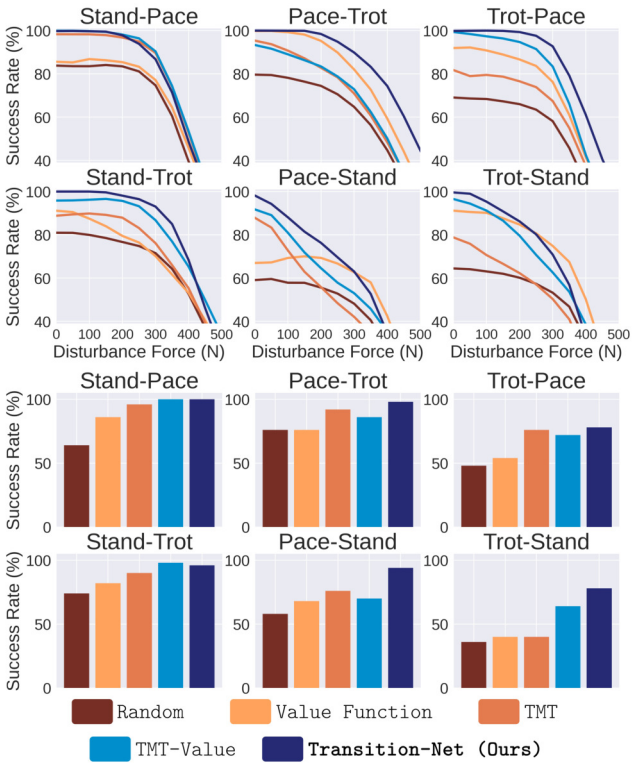

Fig. 4. Impact of varying disturbances to the transition success rate in simulation (top) and real-world transition success rate (bottom).

is trained in the simulation setting and might produce noisy value estimates in the real-world setting. In this experiment we do not introduce external disturbances to the robot, allowing ***TMT*** and ***TMT-value*** to achieve high success rates and showing that phase serves as a reasonably accurate proxy. Finally, using the policy's latent representation with our ***transition-net*** yields the most robust transitions. It can accurately identify the most appropriate configuration to perform the transitions for every pair in the real-world.

Jointly analyzing the results from the simulation and real-world settings, we observe that the most challenging transition are dynamic-to-static transitions, i.e., when the robot performs a full-stop after a moving gait, such as *pace–stand* and *trot–stand*. This is evidenced by the steeper curves in Figure 4 (top) and the low success rate of the ***random*** approach in the real-world in Figure 4 (bottom). The ***transition-net*** achieves 19% higher average success rate compared to the second best strategy, ***TMT-value***.

B. Latent Representation of Agent State

To investigate the effectiveness of using the latent state representations ψ_m as the input of *transition-net*, we experiment with two alternative representations: phase variable ϕ_m and the explicit state of the agent, a subset of $\boldsymbol{s}_t$. For each representation, we measure the accuracy of the *transition-net* in predicting successful transition configurations given a balanced validation set. The dataset contains 5500 samples for each pair, with a total of 33000 samples. The experiment

TABLE III

EFFECTS OF DIFFERENT STATE REPRESENTATIONS FOR THE TRANSITION-NET INCLUDING (S)TAND, (P)ACE, AND (T)ROT GAITS.

Representation	Accuracy (%)					
	S-P	S-T	P-T	P-S	T-P	T-S
Phase (ϕ_m)	88.94	**87.59**	63.86	67.42	66.14	65.01
Explicit ($\boldsymbol{s}_t$)	**89.15**	87.5	81.26	78.01	76.16	71.85
Latent (ψ_m)	87.14	86.97	**82.45**	**78.68**	**78.3**	**75.49**

results shown in Table III illustrate the strengths of each representation.

The phase variable remains a solid low-dimensional proxy for the agent's state when the source policy is not very dynamic. This behavior is highlighted by the high accuracy in transitions where *stand* is the source policy, but poor accuracy for all other transitions. A more descriptive state representation is to explicitly use the recent history of the agent's joint positions, orientation, angular velocity, and feet contacts. With this input, the *transition-net* can internally estimate the phase information while having a more comprehensive state representation, resulting in higher accuracy than simply using the phase variable. In the end, the latent state representation encapsulates all of these desirable properties with the addition of implicit encoding of the environment obtained during the training process in simulation. Therefore it yields the highest prediction accuracy in transitions involving more dynamic states, such as the ones originating from *pace* and *trot*.

VI. CONCLUSION

We proposed a robust transition strategy that enables a meta-controller to expand the versatility of locomotion in a quadruped robot. Containing the complexity of each gait in independent policies is pivotal in making the training and sim-to-real process more tractable. The experiments demonstrated that the latent state representation served as a reliable proxy for the state of the robot and its environment dynamics. It enabled the *transition-net* to robustly identify successful transition configurations. Our setup of independent policies paired with the meta-controller enables iterative skill expansion while preserving learned ones. Finally, our massively parallel implementation allowed us to extensively iterate through policies and design choices.

In the future, we plan to perform experiments with a larger number of motion policies to further validate the iterative expansion property of our method. Additionally, we want to investigate more refined mechanisms of policy switching, such as generating transition trajectories. Doing so, would enable transitions between policies that are very distinct. We believe our work opens up interesting research directions toward versatile and robust legged robots.

ACKNOWLEDGEMENT

We wish to thank Trista Chen for the meaningful discussions that lead to the design choices of our proposed work and our colleagues from the Inventec AI Center for their help in producing the supplementary video.

REFERENCES

[1] Z. Fu, A. Kumar, J. Malik, and D. Pathak, "Minimizing energy consumption leads to the emergence of gaits in legged robots," *Conference on Robot Learning (CoRL)*, 2021.

[2] A. Iscen, G. Yu, A. Escontrela, D. Jain, J. Tan, and K. Caluwaerts, "Learning agile locomotion skills with a mentor," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2019–2025.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] Y.-S. Luo, J. H. Soeseno, T. P.-C. Chen, and W.-C. Chen, "Carl: Controllable agent with reinforcement learning for quadruped locomotion," *ACM Trans. Graph.*, vol. 39, no. 4, 2020.

[8] 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.

[9] 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.

[10] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," *ACM Trans. Graph.*, vol. 36, no. 4, p. 42, 2017.

[11] 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.

[12] 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.

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

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] S. Nansai, N. Rojas, M. R. Elara, R. Sosa, and M. Iwase, "A novel approach to gait synchronization and transition for reconfigurable walking platforms," *Digital Communications and Networks*, vol. 1, no. 2, pp. 141–151, 2015.

[26] 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.

[27] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, 2021.

[28] 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.

[29] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," in *Robotics: Science and Systems*, 2021.

[30] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan, "Learning to utilize shaping rewards: A new approach of reward shaping," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 931–15 941, 2020.

[31] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, "Policies modulating trajectory generators," in *Conference on Robot Learning*. PMLR, 2018, pp. 916–926.

[32] 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.

[33] L.-K. Ma, Z. Yang, X. Tong, B. Guo, and K. Yin, "Learning and exploring motor skills with spacetime bounds," in *Computer Graphics Forum*, vol. 40, no. 2. Wiley Online Library, 2021, pp. 251–263.

[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, pp. 33–1, 2020.

[35] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions," *arXiv preprint arXiv:2203.15103*, 2022.

[36] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[37] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *Robotics: Science and Systems (RSS)*, 2018.

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

[39] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2811–2817.

[40] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7309–7315.

[41] 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.

[42] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.