2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

PM-FSM: Policies Modulating Finite State Machine for Robust Quadrupedal Locomotion

Ren Liu, Nitish Sontakke, Sehoon Ha

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9982259

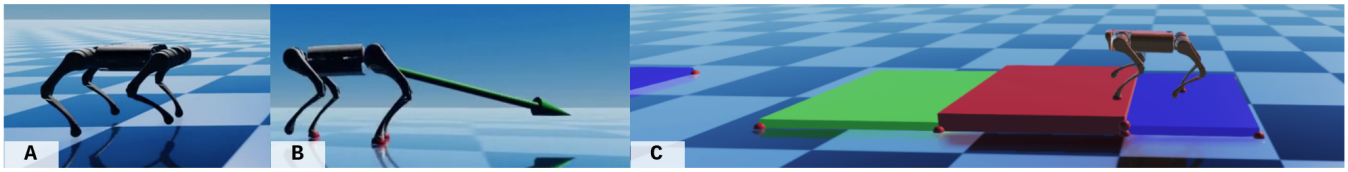


Fig. 1. Locomotion tasks covered in our simulated experiments, including the flat terrain (A), the flat terrain with external perturbations (B) and the randomized upstairs/downstairs terrains (C). Our proposed PM-FSM model proves its robustness on these simulation tasks, as well as real-world experiments.

***Abstract*—Deep reinforcement learning (deep RL) has emerged as an effective tool for developing controllers for legged robots. However, vanilla deep RL often requires a tremendous amount of training samples and is not feasible for achieving robust behaviors. Instead, researchers have investigated a novel policy architecture by incorporating human experts' knowledge, such as Policies Modulating Trajectory Generators (PMTG). This architecture builds a recurrent control loop by combining a parametric trajectory generator (TG) and a feedback policy network to achieve more robust behaviors. In this work, we propose Policies Modulating Finite State Machine (PM-FSM) by replacing TGs with contact-aware finite state machines (FSM), which offers more flexible control of each leg. This invention offers an explicit notion of contact events to the policy to negotiate unexpected perturbations. We demonstrated that the proposed architecture could achieve more robust behaviors in various scenarios, such as challenging terrains or external perturbations, on both simulated and real robots.**

***Index Terms*—Finite State Machine, Reinforcement Learning, Quadrupedal Locomotion**

I. Introduction

Locomotion has been one of the most challenging problems in the robotics domain. A controller must achieve its primary task of moving its body toward the desired direction while maintaining the balance with limited sensing and actuation capabilities. One popular approach is model-based control that leverages identified dynamics and control principles, demonstrating effective locomotion on quadrupedal [7], [20] and bipedal robots [21], [30]. However, model-based control often requires considerable manual effort to develop a proper model for each task. On the other hand, deep reinforcement learning (deep RL) has emerged as a promising approach to learn a robust policy by maximizing the reward [4], [8], [27]. To cope with extrapolated tasks, researchers have investigated more and more complex policies, such as long term short memory [5] or temporal convolutional neural networks [11]. These policies require even more training samples and computations resources to obtain better performance, compared with common feed-forward network architectures.

Researchers have attempted to develop a hybrid technique to take the best from both model-based and learning-based approaches to reach both performance and time efficiency. For instance, a few prior works proposed hierarchical controllers where a policy outputs high-level parameters to low-level model-based controllers that are typically implemented with the Model Predictive Control strategy (MPC) [14], [29], [32]. Although this approach is known to be sample-efficient and robust, its performance highly depends on the design of the low-level controller. Alternatively, researchers have investigated Policies Modulating Trajectory Generators (PMTG) [10], [12], [34] that embeds prior knowledge of trajectory generators into neural network policies and demonstrated great sample efficiencies.

Our key intuition is that the explicit notion of contacts can greatly improve the robustness of control policies [1], [19], just like other model-based controllers. In PMTG, the predefined trajectory generator offers both memory and prior knowledge to the control policy to handle periodic locomotion. Still, it is difficult to learn very discrete behaviors due to the smooth nature of neural networks, even if we provide contact flags as additional inputs. On the other hand, finite state machines (FSM) can be more expressive by modeling abrupt behavior changes conditioned on contact signals.

In this work, we propose a novel policy representation that is explicitly aware of locomotion context including foot contact flags. We propose a novel policy architecture, Policies Modulating Finite State Machine (PM-FSM), which extends the trajectory generators in the original PMTG. Our key idea is to replace trajectory generators with finite state machines. This simple extension allows a robot to be aware of contact events explicitly and effectively adapt its behaviors to perturbed scenarios.

Georgia Institute of Technology, Atlanta, GA, 30308, USA
rliu384@gatech.edu, nitishsontakke@gatech.edu, sehoonha@gatech.edu

We evaluate the proposed PM-FSM on the locomotion task of a quadrupedal robot, A1, in both simulated and real-world environments. Our results suggest that our architecture shows more robust behaviors than the original PMTG and its variant in various scenarios with external perturbations, and is better at the sim-to-real transfer. We then show that some complicated reflexes, such as going upstairs and downstairs, can also be learned using PM-FSM.

II. Related Works

A. Locomotion with FSM

It has been a very long time for researchers to use FSMs to make robots walk. Mcghee *et al.* [17], [18], [28] are some of the earliest studies to prove that simple FSMs can accomplish the coordination of effective joint movements. Mcghee *et al.* [18] defined each leg of the quadrupedal as a two-state automata. They also developed a hierarchical structure where the states of the hip and knee joints act as a sub-automata of the leg belonged to. Since then, more complicated controllers based on finite state machines have been developed for various applications related to robotic locomotion. Park *et al.* [19] adopted the idea of sensory reflex-based control strategy [6] and designed an finite state machine to manage unexpected large ground-height variations in bipedal robot walking. And Lee *et al.* [13] showed that finite state machines can also be used in a data-driven biped control to generate robust locomotion. Recently, Bledt *et al.* [1] showed that locomotion context like contact states of each leg can be used in finite state machines to help to generate stable locomotion and to assist gait switching. However, the design of effective FSMs requires a lot of time-consuming trials and errors based on prior knowledge if the goal is to overcome challenging terrains or dynamic environments.

B. Locomotion with Deep RL

The recent advances of deep reinforcement learning enable a more convenient approach for developing control policies by leveraging simple reward descriptions. Various policy gradient methods, such as Deep Deterministic Policy Gradient (DDPG) [15], Trust Region Policy Optimization (TRPO) [24] and Proximal Policy Optimization (PPO) [25], have been widely adopted to train effective control policies, particularly in the context of robotic locomotion. As these methods usually learn policies in simulated environments, a lot of efforts to overcome the *reality gap* have been made, such as domain randomization [2], [27], [33], meta learning [3], [35], and compact observation space [27]. Recently, a few learning-based approaches including training an extra network to model actuator dynamics [8] or training a Cycle-GAN that maps the images from the simulator to corresponding realistic images [22] have also reached great performance in real robot experiments.

Although simple MLPs have been used as policy network in a large number of studies [2], [27], [33], PMTG [10] offers an effective approach to improve the performance of this simple policy architecture by taking an advantage of pre-defined trajectory generators (TGs). In this architecture, the policy can directly modulate the behavior of the predefined trajectory generators while generating additional feedback control signals. It has been shown that PMTG architecture can produce robust locomotion policies in both simulated and real-world environments.

In this paper, we propose a new policy architecture, PM-FSM, by combining both directions, FSMs and deep RL. The prior knowledge is presented as a contact-aware FSM, of which the state transition functions are defined according to the robot proprioceptive sensory information (foot contact flags and target joint angles). This FSM is modulated with a feedback network which learns a robust control policy using deep RL.

III. Policies Modulating Finite State Machine

A. Background: PMTG

PMTG [10] divides policy outputs into the trajectory generator modulating parameters (ρ) and the feedback terms (u_{fb}). The policy modulating parameters ρ include the trajectory generator (TG)'s frequency f, amplitude A, and height h. Then the TG converts the parameters into control signals u_{tg} and add them with the feedback terms. This formulation allows a policy to explore smooth behaviors by leveraging the pre-defined TG, while fine-tuning the behaviors with the feedback terms. Typically, PMTG is optimized with on-policy RL algorithms, such as PPO [25] and ARS [16]. For more details, please refer to the original paper.

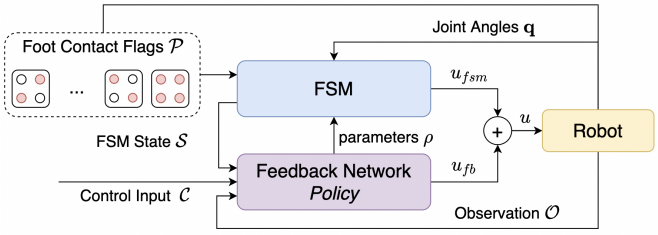


Fig. 2. Overview of PM-FSM: The output (actions) u_{fsm} of a predefined FSM combined with that of a learned feedback network (u_{fb}). The learned policy also modulates the parameters ρ of the FSM at each time step and observes its state $\mathcal{S}$. The state transition conditions of the FSM are based on the current contact flags $\mathcal{P}$ and joint angles $\mathbf{q}$.

B. Overview of PM-FSM

Our key idea is to further extend the TG of the PMTG with an FSM that is explicitly aware of contacts. The top-level architecture for PM-FSM is shown in Figure 2. Compared with PMTG, our FSM explicitly considers locomotion contexts Σ that consists of the current foot contact flags $\mathcal{P}$ and joint angles $\mathbf{q}$. Note that these locomotion contexts are directly read from the robot sensors and does not need to be part of the policy inputs, although they often overlap in practice.

C. Policy Modulating Parameters

In our system, the policy takes FSM state $\mathcal{S}$ and generates the policy modulating term $\rho = (f, A, h)$, frequency f, amplitude A, and height, along with the feedback term u_{fb}).

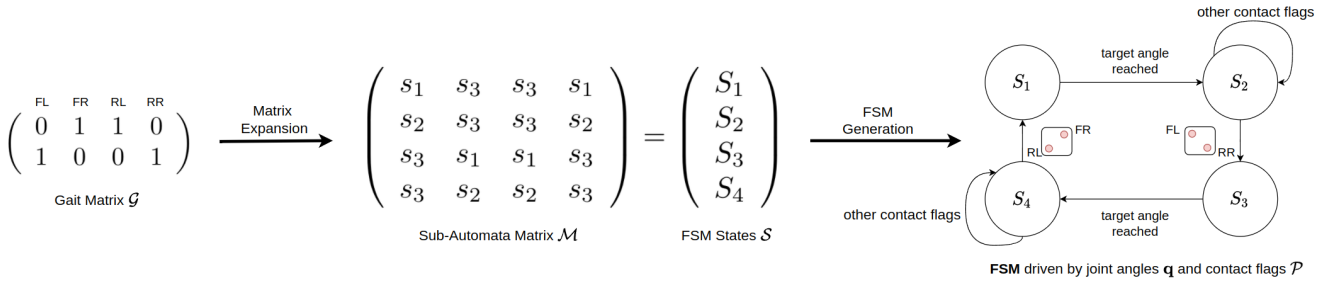


Fig. 3. Our Finite State Machine Design. We expand the given gait matrix $\mathcal{G}$ by dividing the transfer phase (0) to leg extension (s_1) and leg retraction (s_2) states. We keep the support phase (1) as leg angle adjustment state (s_3). Then every row in the sub-automata matrix $\mathcal{M}$ forms an FSM state. FL: Front Left, FR: Front Right, RL: Rear Left, RR: Rear Right.

Then FSM combines these parameters and locomotion contexts $\Sigma = (\mathcal{P}, \mathbf{q})$ to configure each leg motion generator and predict next actions u_{fsm}. Note that there are no learnable parameters within the FSM, which is similar to PMTG. FSM changes its behavior only via policy modulation.

In PMTG, the frequency f_{tg} will influence the speed of joint rotation. As FSM controller does not have a shared system cycle time, we use f to determine the ideal cycle step T when the robot walks on the flat terrain without external perturbations. With control time interval as dt, we define the ideal cycle step T as:

$$T = \lceil \frac{1}{f \cdot dt} \rceil. \tag{1}$$

The amplitude A and height h will decide the target joint angles, which does not require a well-tuned configuration.

D. Design of Finite State Machine

In this section, we will describe the gait-based FSM (Figure 3). Our design is inspired by a gait matrix $\mathcal{G}$ introduced by Mcghee *et al.* [18]. In their definition, each leg of the robot can be designed as a 2-state automata in locomotion. These two states are usually referred to as the *support phase* and *transfer phase*, which represent the state that supporting on the ground and swinging forehead separately. And a gait matrix $\mathcal{G}$ with k columns is introduced to describe the locomotion automation composed of k legs, where $\mathcal{G}_{ij}$ means in the gait state i, whether the leg j is in support ($\mathcal{G}_{ij} = 0$) or transfer ($\mathcal{G}_{ij} = 1$) phase.

We further build a sub-automata matrix $\mathcal{M}$ by expanding the given gait matrix $\mathcal{G}$ by adopting a three-state joint sub-automata with the Swing Retraction model [26], [31]. In this purpose, we divide the transfer phase (0) into leg extension (s_1) and a leg retraction (s_2) sub-automata states. Note that this method requires that there is no successive transfer phases in any column of the gait matrix $\mathcal{G}$. Then we can compute joint angles for each sub-automata states s_1, s_2, and s_3 using the FSM parameters (f, A, h) given from a policy: please refer to Figure 4 for more details. This expansion allows us a finer control over each leg.

Then we build the final FSM states $\mathcal{S}_i$ from each row of the expanded automata $\mathcal{M}$. The start state of the FSM is set to $\mathcal{S}_1$ in default. The set of final states $\mathcal{F} = \phi$ because we aim to develop an indefinite walking controller. Finally, we define the state transition functions as:

$$\delta(\mathcal{S}_i, \mathcal{S}_{i+1}) = \begin{cases} \text{all swing legs make contacts, if } \mathcal{S}_2 \text{ or } \mathcal{S}_4 \\ \text{all joints reach target angles, if } \mathcal{S}_1 \text{ or } \mathcal{S}_3 \end{cases}$$

Then this formulation defines the FSM as quintuple $(\Sigma, \mathcal{S}_1, \mathcal{S}, \delta, \mathcal{F})$.

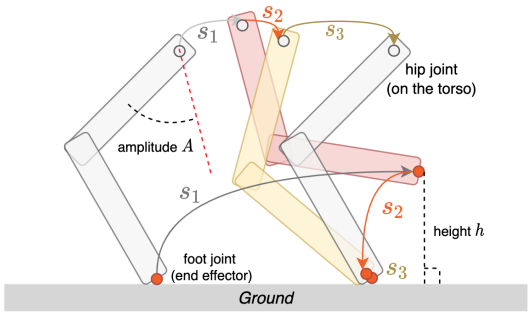


Fig. 4. An Example of Joint Sub-Automata: **Leg extension[s_1]:** The leg moves from the most rear position to the most front position. **Leg retraction[s_2]:** The raised leg will move back to contact with the ground. **Leg angle movements[s_3]:** The foot joint will be kept on the ground while moving the hip/knee angles to lift and push forward the torso. The amplitude A defines the targe hip angle difference between s_1 and s_2, and the height h defines the distance from the lifted foot to the ground at the end of $s1$.

E. Reflexes

The explicit notion of contact flags also allows us to easily extend PM-FSM with reflex controllers. Park *et al.* [19] proposed the idea of tripping reflex as part of the FSM to cope with complex tasks like going upstairs and downstairs. These model-based methods often require precise tuning to maximize their performance. However, in our framework, we expect a feedback policy is capable of learning how to leverage heuristic reflex controllers.

There are two simple reflexes designed in our FSM: going upstairs and downstairs reflexes. To activate the upstairs reflex (UR), we detect if there is an unexpected contact signal before the leg reaches its target joint angle during the leg extension state (s_1). To activate the downstairs reflex (DR), we detect whether the leg is dangling when the leg reaches its target joint angle at the end of the leg retraction state (s_2). In each case, we propose two-step and one-step reflex

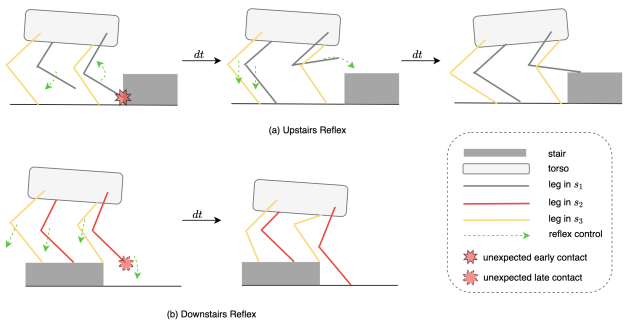


Fig. 5. Upstairs reflex (UR) and downstairs reflex (DR) in gait-based FSMs: (a) UR: When unexpected early contact happens while the leg is in the leg extension state (s_1), first raise the collided leg higher and retract other extended legs, then bend the rear legs (RL, RR) and retry leg extension. (b) DR: When unexpected late contact happens while the leg is in the leg retraction state (s_2), extend the dangling leg and lower the torso.

mechanisms for UR and DR, respectively. See Figure 5 for details.

Note that all parameters in these reflexes, including how much the collided leg will be lifted and how much the rear legs will bend, are determined heuristically without fine-tuning. These parameters are expected to be further adjusted by the feedback network. To make the policy be aware of applied reflexes, an additional *reflex state* showing the current activated reflex's ID will be appended to FSM state $\mathcal{S}$ to be passed to the feedback network.

F. Interpolated Motion Control

In our design, we parameterize joint sub-automata with three target poses with six variables. When we want to control the robot, we do not want to use the given pose directly as PD target because it will cause unnecessarily abrupt movements. Instead, our goal is to apply exponential interpolation for achieving smoother motions by designing the following kinematic controller. For a k-state finite state machine, the duration distribution τ is defined by a k-dim vector

$$\tau = [d_1, ..., d_k], \text{ where } \sum_{i=0}^{k} d_i = 1.$$

and the termination position for each state is defined by the k-tuple of vectors $\epsilon = (\mathbf{e}_1, ..., \mathbf{e}_k)$.

Let us assume that the current state is i, the current time is t steps after entering the state i, and the current position is $\mathbf{p}(t)$, a simple exponential model is adopted to calculate the expected action $\mathbf{a}_t$, which suggests the increment from the current joint angles to the next angles:

$$\mathbf{a}(t) = K_i \cdot (\mathbf{e}_i - \mathbf{p}(t)), \tag{2}$$

where K_i is a coefficient. We can solve K_i to achieve the desired durations:

$$K_i = 1 - \sqrt[\lfloor T \cdot d_i + 0.5 \rfloor]{\frac{\delta}{|e_i - e_{i-1}|}} \tag{3}$$

where T is the total steps in the ideal situation and δ is the toleration.

In our joint sub-automata, the duration distribution is set as a constant vector $\tau = [0.34, 0.16, 0.50]$, which is similar to the $1:2$ duration distribution for *transfer phase* and *support phase* mentioned in the work of Mcghee *et al.* [17].

IV. Experiments

We conducted both simulated and real robot experiments to show that our PM-FSM allows us to learn robust quadrupedal locomotion policies on challenging terrains. Particularly, we designed our experiments to verify the following research questions:

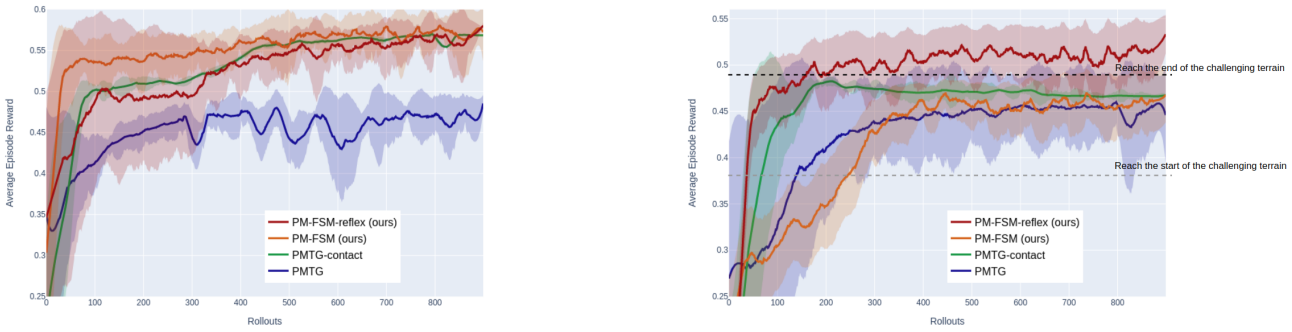
1) Can PM-FSM policies show more robust behaviors than PMTG policies?
2) How do different design choices (like reflexes) affect the performance of PM-FSM?

A. Simulated Experiments

Experimental Setup. We took the RaiSim [9] as our training environment. The friction coefficient of the training terrains is set to 0.8 to better simulate the blanket floor. We also added random directional perturbation forces, up to $10N$ horizontally and $30N$ vertically, to the robot multiple times at random timing to obtain more stable behaviors. To fit the given specification of AlienGo and A1 Explorer robots from Unitree [23], we set the maximum velocity in the task v_{max} as 0.6 m/s. The control frequency is 40 Hz and each rollout lasts for 25 seconds. The total number of roll-out is set to 2000.

We learned a policy with an Actor-Critic on-policy learning method, Proximal Policy Optimization. Both we represent the actor and critic networks as simple three-layer fully connected neural networks. The numbers of hidden nodes in two hidden layers are 128 and 64 respectively. We trained four policies for each experiment: 1)PMTG, 2)PMTG-contact, 3)PM-FSM, 4)PM-FSM-reflex. We did not compare them with a naive reactive agent without prior knowledge because they tend to require a far more simulation steps. All policies take z-axis of the robot frame in the world frame (3-dim), the current linear velocity of the robot along the target direction (1-dim), the current angular velocities of the robot (roll, pitch, yaw; 3-dim), and the target velocity generated by the velocity controller (1-dim) as input. Additionally, PMTG and PMTG-contact take a 2-dim embedding of TG phase [10], and PMTG-contact takes extra 4-dim contact flags of each foot. Our PM-FSM and PM-FSM-Reflexes take the 4-dim sub-automata states for all legs, and PM-FSM-reflex takes an extra 1-dim reflex state as input FSM state. The output dimension is 11 for all cases: 8 dimensional u_{fb} for pitch joint angles and 3 dimensional (f, A, h) for modulating parameters ρ.

Our reward function is designed to be sum of three terms: r_{speed}, r_{torque} and r_{done}. The speed term r_{speed} measures the difference between the current and target velocities, which is



(a) learning on flat terrain (with random perturbation)

(b) learning on challenging terrain (with random perturbation)

Fig. 6. Learning curves: (a) All PM-FSM-reflex (Ours), PM-FSM (Ours), and PMTG-contact learned a stable policy in the *training* environment except for the vanilla PMTG without contact flags. (b) However, only PM-FSM-reflex agent learned an effective policy to go through the whole challenging scenarios with stairs and random perturbations.

exactly the same as the reward defined in the original PMTG paper:

$$r_{speed} = v_{max} \exp(-\frac{(v_R - v_T)^2}{2v_{max}^2}), \tag{4}$$

where v_{max} is the maximum desired velocity for the task, and v_R and v_T are the robot's actual velocity and the target velocity at the current timestep. The torque term r_{torque} is simply a negative coefficient C_τ times the squared sum of torques $\|\tau\|^2$. The termination penalty r_{done} is -0.5 if the robot falls before the end of the rollout and is 0 otherwise.

We designed three simulated experiments (Figure 1) to explore the answer to the research questions proposed. In each experiment, we evaluate the learned policies on the following tasks:

1) **VEL**: Random velocity profiles ($v_{max} \in [0.5, 0.9]$ m/s)
2) **PER**: Random perturbations ($F_z \leq 20$N, $F_{xy} \leq 10$N)
3) **STR**: Random stairs ($\Delta_{altitude} \leq 5$ cm, length ≤ 1 m)

where F_z, F_{xy} are vertical and horizontal external forces on the torso. $\Delta_{altitude}$ is the maximum altitude change between neighbor stairs. The first two experiments share the same policies trained on the solely flat terrain, while policies for the last experiment are trained on challenging terrains described in Figure 7. All experiments last for 25 seconds. These three experiments are designed for three types of scenarios separately: a flat terrain without perturbations (*Flat/Stationary*), a flat terrain with perturbations (*Flat/Perturb*) and a random terrain without perturbations (*Random/Stationary*).

We evaluate three tasks based on the following criteria. For **VEL** test, we use average velocity error $\bar{E}_v$ to evaluate whether agents are capable of following different velocity profiles. For **PER** and **STR** tests, we use average travel distance $\bar{D}$ to evaluate whether agents are able to keep stable locomotion under unexpected perturbations or obstacles. The expected travel distance was 15.0 m for both tests.

Results. First, we compared the learning curves in two different training terrains of four agents, PMTG, PMTG-contact, PM-FSM (ours), and PM-FSM-reflex (ours), in Figure 6. The results indicate that PM-FSM and PMTG-contact agents achieved much higher rewards than PMTG on the flat terrain by closely following the desired velocity profile. PM-FSM agents have even larger rewards ($\geq 0.58/0.6$) than PMTG-contact. Our PM-FSM-reflex is the best agent in both challenging tasks, **PER** and **STR**. For all stair-specific training, our PM-FSM-reflex is the only one that learns effective policies to go through the whole challenging scenarios with stairs and random perturbations.

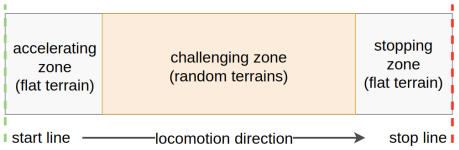


Fig. 7. Training and testing environment for challenging terrains: simulated robot needs to accelerate to the target speed on a flat terrain, then go through a challenging zone with stairs and finally slow down to stop. The width of stairs are all equal to the width of the terrain and the maximum altitude change is 5 cm for each stair.

We then evaluated the robustness of all the three agents by deploying them to scenarios that are unseen during training. We conducted ten tests by randomly generating velocity profiles, external perturbations, and stair configurations.

TABLE I
RESULTS OF SIMULATED TEST EXPERIMENTS

Experiment	**VEL**	**PER**	**STR**
Scenarios	Flat/Stationary	Flat/Perturb	Random/Stationary
Measurements	$\bar{E}_v$ $(m/s)\downarrow$	$\bar{D}$ $(m)\uparrow$	$\bar{D}$ $(m)\uparrow$
PMTG [10]	0.254	7.38	0.84
PMTG-contact	0.115	9.20	1.25
PM-FSM	**0.055**	8.35	1.71
PM-FSM-reflex	0.064	**10.25**	**5.34**

Table I presented the comparison results obtained in the tested unseen scenarios. For the **VEL** task, the results show both PM-FSM agents were able to produce velocities closely according to the input target velocities, while both PMTG and PMTG-contact agents resulted in much larger velocity errors. This trend indicates that PM-FSM performs more robust on unseen tasks because the training performances of PMTG-contact, PM-FSM, and PM-FSM-reflexes are comparable in Figure 6. For the **PER** test, PM-FSM-reflex agent performed the best to resist the impacts from random perturbations. For the **STR** scenario, PM-FSM with reflexes was the only agent that managed to go through most random stair terrains.

These results support our hypothesis that the explicit notion of contact flags (PM-FSM, PM-FSM-reflex) is helpful to obtain more robust agents compared to the zero awareness of contacts (PMTG) or the naive formulation (PMTG-contact). The results also suggest that the designed reflex mechanism helped the agent to overcome challenging terrains as well as obtain more robust policies.

For a more detailed analysis, we plotted the foot joint trajectory of the front right leg (FR) in both flat and challenging terrains to visualize how the PM-FSM-reflex agent managed to overcome stairs (Figure 8). We also plotted the corresponding trajectories using an untrained PM-FSM-reflex and a trained non-reflex PM-FSM. We observed that non-reflex PM-FSM was unaware of heights and only learned to keep balance. Although the untrained PM-FSM-reflex successfully put the FR leg on the stair, it failed to lift the whole body onto the stairs. On the other hand, PM-FSM-reflex learned to put the foot joint a little backward to raise its torso and go through the stairs. This result supports the claim that our PM-FSM-reflex well learns to modulate the behaviors of the reflexes to overcome various stairs.

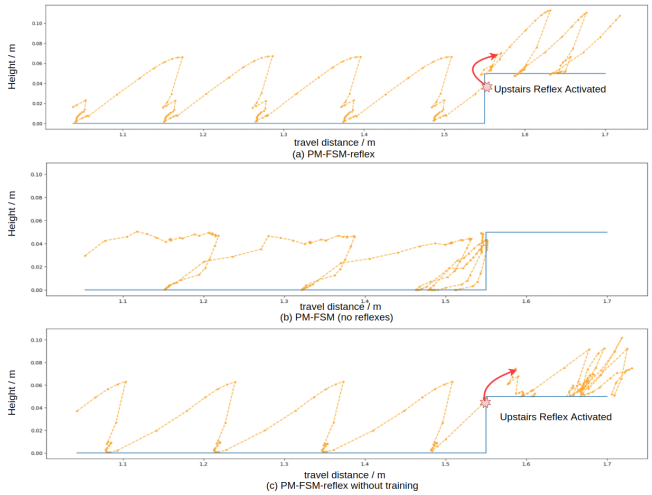


Fig. 8. FR foot joint trajectories: (a) PM-FSM-reflex, (b) PM-FSM, (c) PM-FSM without training. We use points to track the foot joint positions every control interval, and use dashed lines to show approximate trajectories between control intervals.

B. Real Robot Experiments

We also deployed the policy learned in the simulated environments to the real A1 robot to test whether our approach is robust enough to handle the sim-to-real gap. The main challenges for the sim-to-real transfer are control latency and observation noise. To deal with the challenges, we import domain randomization in the training process by randomizing some "missing" control steps, randomizing the position of the center of mass, and randomizing the external perturbation. Both policies are trained under exactly the same conditions (number of total rollouts, random seed, etc.). In the real robot experiments, we set the test maximum velocity at 0.5 m/s with a control frequency of 20 Hz. Figure 9 shows the comparison between the real robot experiments and the corresponding simulated experiments. From the results, we

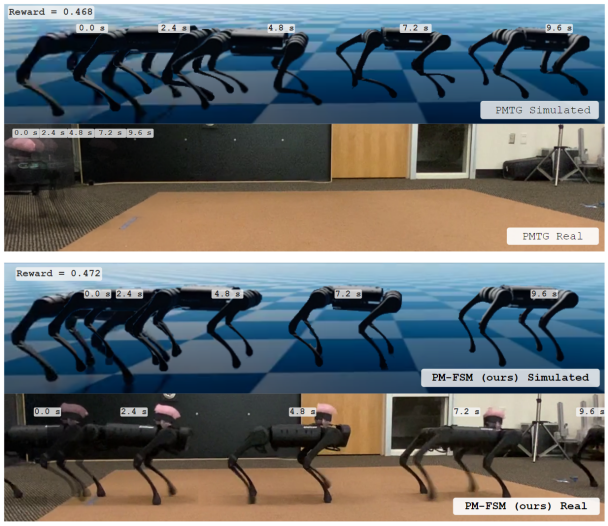


Fig. 9. Result of simulated and real robot experiments. Time tickets at $0.0s$, $2.4s$, $4.8s$, $7.2s$, and $9.6s$ are chosen for recording the robot's positions.

observed that the original PMTG model failed to produce a successful forwarding policy. And our PM-FSM model moved forward to the target, for contact-aware FSM controller can be more robust against unpredictable mechanical latency.

Please note that this experiment does not disprove the sim-to-real transferability of the original PMTG. If we use more complex training approaches or carefully tune DR parameters, it would be possible to obtain a reasonable PMTG policy that works on the hardware. However, we observed that our PM-FSM could better bridge the sim-to-real gap without careful domain randomization tuning.

V. Conclusion

In this work, we propose a novel policy architecture, Policies Modulating Finite State Machines (PM-FSM), that takes advantage of the prior knowledge of locomotion contexts and the associated walking automata. Our key idea is to extend the trajectory generator with a finite state machine, as its name suggests. This invention allows us to learn a robust locomotion policy by being explicitly aware of the desired event sequences. We evaluate the proposed method on various challenging locomotion tasks with randomized terrains and perturbations. We demonstrate that our method can outperform two baselines: PMTG and PMTG with contact flags, which demonstrates the importance of the explicit awareness of contacts. We also succeed in the sim-to-real transfer from the RaiSim environment to the A1 robot.

Based on our current results, we plan to extend the current FSM design to learn more robust quadrupedal locomotion. In this work, our contact-aware FSM is based on simple ideas proposed by Mcghee [18]. In future work, we are looking forwards to importing advanced design of FSM-based locomotion controllers to our architecture as more powerful modules. We believe this extension will give a policy the ability to sense a broad range of events more explicitly, eventually leading to better robust behaviors in many challenging environments.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation under Award #2024768. We would like to thank Visak C.V. Kumar for his help during the real world deployment.

REFERENCES

[1] Gerardo Bledt, Patrick M Wensing, Sam Ingersoll, and Sangbae Kim. Contact model fusion for event-based locomotion in unstructured terrains. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4399–4406. IEEE, 2018.

[2] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience, 2019.

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[4] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] Qiang Huang and Yoshihiko Nakamura. Sensory reflex control for humanoid walking. *IEEE Transactions on Robotics*, 21(5):977–984, 2005.

[7] Marco Hutter, Hannes Sommer, Christian Gehring, Mark Hoepflinger, Michael Bloesch, and Roland Siegwart. Quadrupedal locomotion using hierarchical operational space control. *The International Journal of Robotics Research*, 33(8):1047–1062, 2014.

[8] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, Jan 2019.

[9] Jemin Hwangbo, Joonho Lee, and Marco Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018.

[10] Atil Iscen, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. Policies modulating trajectory generators, 2019.

[11] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017.

[12] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[13] Yoonsang Lee, Sungeun Kim, and Jehee Lee. Data-driven biped control. In *ACM SIGGRAPH 2010 papers*, pages 1–8. 2010.

[14] Chao Li, Xin Min, Shouqian Sun, Wenqian Lin, and Zhichuan Tang. Deepgait: a learning deep convolutional representation for view-invariant gait recognition using joint bayesian. *Applied Sciences*, 7(3):210, 2017.

[15] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

[16] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

[17] Robert B McGhee. Finite state control of quadruped locomotion. *Simulation*, 9(3):135–140, 1967.

[18] Robert B McGhee. Some finite state aspects of legged locomotion. *Mathematical Biosciences*, 2(1-2):67–84, 1968.

[19] Hae-Won Park, Alireza Ramezani, and Jessy W Grizzle. A finite-state machine for accommodating unexpected large ground-height variations in bipedal robot walking. *IEEE Transactions on Robotics*, 29(2):331–345, 2012.

[20] Hae-Won Park, Patrick M Wensing, and Sangbae Kim. High-speed bounding with the mit cheetah 2: Control design and experiments. *The International Journal of Robotics Research*, 36(2):167–192, 2017.

[21] Alireza Ramezani, Jonathan W Hurst, Kaveh Akbari Hamed, and Jessy W Grizzle. Performance analysis and feedback control of atrias, a three-dimensional bipedal robot. *Journal of Dynamic Systems, Measurement, and Control*, 136(2):021012, 2014.

[22] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real, 2020.

[23] Unitree Robotics. Unitree official website: https://www.unitree.com/.

[24] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.

[25] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[26] André Seyfarth, Hartmut Geyer, and Hugh Herr. Swing-leg retraction: a simple control model for stable running. *Journal of Experimental Biology*, 206(15):2547–2555, 2003.

[27] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots, 2018.

[28] R. Tomovic and R.B. McGhee. A finite state approach to the synthesis of bioengineering control systems. *IEEE Transactions on Human Factors in Electronics*, HFE-7(2):65–69, 1966.

[29] Vassilios Tsounis, Mitja Alge, Joonho Lee, Farbod Farshidian, and Marco Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3699–3706, 2020.

[30] Eric R Westervelt, Jessy W Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2018.

[31] Martijn Wisse, Christopher G Atkeson, and Daniel K Kloimwieder. Swing leg retraction helps biped walking stability. In *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, pages 295–300. IEEE, 2005.

[32] Zhaoming Xie, Xingye Da, Buck Babich, Animesh Garg, and Michiel van de Panne. Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model. *arXiv preprint:2104.09771*, 2021.

[33] Zhaoming Xie, Xingye Da, Michiel van de Panne, Buck Babich, and Animesh Garg. Dynamics randomization revisited: A case study for quadrupedal locomotion. *arXiv preprint arXiv:2011.02404*, 2020.

[34] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pages 1–10. PMLR, 2020.

[35] Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization, 2020.