Reinforcement Learning Based Legged Locomotion of Quadrupedal in a Complex Environment

Anurup Atul Salokhe
School of Computer Science and Engineering
Vellore Institute of Technology (VIT)
Vellore, Tamil Nadu, India
anurup.salokhe08@gmail.com

Dr. Murugan Krishnamoorthy
School of Computer Science and Engineering
Vellore Institute of Technology (VIT)
Vellore, Tamil Nadu, India
murugan.k@vit.ac.in

***Abstract*—In this work, we propose a Reinforcement learning based legged locomotion quadruped comprising active model that learns to localize itself in a space known by the system in complex cluttered environment. Legged robots, like quadrupeds, would be introduced in terrains with difficult-to-model and predict geometries, so they must be fitted with the ability to generalize well to unexpected circumstances, according to the problem statement. We proposed a novel method for training neural-network policies towards terrain-aware functionality in this paper, which incorporates state-of-the-art frameworks based-model for robot navigation and reinforcement learning. Our strategy is as follows instead of using physical simulation, we formulate Markov decision processes (MDP) based on the assessment of complex feasibility parameters. Using both exteroceptive and proprioceptive measurements, we employ policy-gradient approaches that develop policies that plan and execute foothold and baseline movements in a variety of environments independently. We put our method to the test on a difficult range of virtual terrain scenarios that include features like narrow bridges, holes, and stepping stones, as well as train policies that successfully locomotive in all situations. The simulation can be achieved on high performing CPU system using Open AI GYM and also using TensorFlow on Rex GYM environment tool.**

Keywords—Reinforcement Learning (RF), Legged Locomotion, MDP, Quadrupedal, Open AI GYM

I. INTRODUCTION

The problem of engineering agile locomotion for quadruped robots has been studied for several years. This is because controlling an under-actuated robot performing highly dynamic motions that require intricate balance is difficult. Classical methods often necessitate a great deal of practice and time-consuming manual tuning. Is it possible to automate this procedure? Deep reinforcement learning has made considerable strides recently. These algorithms are capable of solving locomotion problems without the need for much human interaction. However, the majority of these experiments are done in simulation, and a controller trained in simulation often fails in the real world. Model inconsistencies between the virtual and actual physical systems create this reality difference. This discrepancy is caused by a variety of factors, including unmodeled dynamics, incorrect simulation parameters, and numerical errors. Worse, this gap is intensified in locomotion tasks. The transitions of touch circumstances split the control space into broken fragments while a robot moves quickly with regular contact changes. Any small model difference can be amplified, resulting in bifurcated outcomes. It's difficult to bridge the perception gap. Another choice is to learn the task on the physical structure directly. Although this has been demonstrated successfully in robotic gripping, it is difficult to apply this approach to locomotion tasks due to the challenges of automatically reconfiguring the experiments and gathering data on a continuous basis. Furthermore, any fall while learning has the potential to harm the robot. Learning in simulation is therefore more attractive for locomotion activities because it is quicker, cheaper, and safer. We present a comprehensive learning framework for agile locomotion in this project, in which control policies are developed in simulation and implemented on real robots. There are two main challenges:

1) Learning locomotion policies that can be regulated.
2) The policies are being transferred to the physical system.

On a quadruped robot, we test our method with two locomotion tasks: tossing and galloping. We demonstrate that highly agile locomotion gaits can emerge automatically using deep RL. We also show how our framework allows users to easily select the locomotion style. When we compare our learned gaits to expert-crafted gaits at the same running pace, we discover that our learned gaits are much more energy efficient. We show how we can effectively execute policies trained in simulation to the physical system using an effective dynamics simulation and robust control policies. The main contributions of this project are:

1) A full learning framework for agile locomotion is proposed. It gives users complete control over the learned policies, ranging from completely restricted toward a user-specified gait to completely trained from scratch.
2) We demonstrate how a number of methods can be used to close the fact gap and perform systematic tests of their efficacy.
3) We show that agile locomotion gaits like trotting and galloping can be trained automatically, and therefore these gaits can be used on robotic systems without any additional physical model training.

II. LITERATURE SURVEY

First, in robotics leg locomotion, both organized and unstructured, in non-flat terrain is a major challenge. In such settings, autonomous operation necessitates highlighting multi-contact motion preparation concerns and management. ANYmal, for example, is a four-legged robot [1], must be able to pick suitable footholds about the terrain though maintaining a constant sense of balance in order to navigate complex environments autonomously. This research focuses

on the issue of using proprioceptive and exteroceptive detecting to prepare and execute foothold sequences in restricted non-flat terrain allowing quadrupedal mobility.

Walking dynamically on non-flat geography, computation of both continuous state-input pathways, such as in the motion, is needed of the foundation, and differentiated decision variables, including which surface to make contact with and when. Model-based methods, even those that combine other heuristics with probabilistic optimization methods [2], [3] [4], have been used to devise motions for both foundation and the feet. While a few of the previously described methods [2], [3] can solve these issues for both discrete and continuous variables, they are still too much time consuming to be used on the platform. Thus, Kino static way of approach [4], [5] had thus far been effective in performing online foothold preparation. In attempt to make the quest for important footholds easier, most methods use some form of terrain parameterization or classification [4], [6], [7].

Attempting to deal with the stochastic problem that emerges from the large number of possible terrain contact configurations, is one of the most difficult aspects of multi-hold planning for multi-limb structures. Using sampling-based analytic techniques [7], [8] or assuming the movement patterns [4], [5] are two common solutions. There are also works [5], [6], [9] that incorporate optimization and random samples approaches. However, since they appear to ignore the dynamics of the system, these usually decouple the acquisition of footholds from the localization of ground movements, remaining Kino static.

Machine-learning methods have also been used in some works to aid terrain descriptions [8], [10], [11]. Others have used Reinforcement Learning (RL) [12], [13] to achieve locomotion that is conscious of the terrain from start to finish. However, using the latter also comes with a number of drawbacks, including: (1) How to remove unwanted emergent behavior while retaining emergent activity that is advantageous and (2) Decrease the total complexion of the study and efficiently trained policies.

We suggest a new approach for to cross complicated non-flat terrain, quadrupedal frameworks are used that incorporates model-free and state-of-the-art based model strategy. Our formulation consisting of: (1) Aware terrain coordinator that produces foothold and base motion sequences that guide the robot in the direction of a goal approaching and (2) A foot base hold and framework acceleration control system that carries out the above sequences also preserving equilibrium and coping through disturbances. Also the planning and the controller were implemented as of deterministic policies that are determined using NN function approximation and programmed using cutting-edge Deep Reinforcement Learning (DRL) algorithms.

A new approach for training Kino dynamic movement planners that uses a Trajectory Optimization (TO) technique to determine the feasibility of transitioning between separate support phases that used a coarse simulation environment. This eliminates the need to have a planner system to communicate with both mobility and a controller during preparation, allowing the two policies to learned separately and reducing overall elemental complexity significantly. The easy formula for creating a dynamic gait control system that depend solely on proprioceptive sensing and use target footholds as references. This allows us to build a controller system that fully utilizes the robot's dynamic behavior to monitor arbitrary target footholds, regardless of the planner that produced them.

Using a physics simulator, we test the efficiency of our system in a variety of difficult locomotion scenarios and report the results. Our experiments indicated that the planner is capable of generalizing over a variety of terrain types and that the control system can track relative footholds while keeping the robot balanced. Furthermore, we compare our system to a state-of-the-art based model approaching [4] to demonstrate its benefits.

III. Methodology

Prior to RL, an agent could learn an optimal strategy for obtaining the maximum return (i.e., the number of rewards) from a setting. The following tuple is used to represent the Markov decision process (MDP): (S, A, R, p_0, p_T, γ). The MDP method can be summarized as follows: The agent begins by randomly obtaining the initial state $s_0 \in S : s_0 \sim p_0 (s_0)$. The agent chooses action $a_t \in A$ by the policy π above the present state $s_t \in S$: $a_t \sim \pi(a_t \mid s_t)$ at time stage $t \in$ N. According to the transfer probability model $p_T : s_t{+}1 \sim p_T (s_{t+1} \mid s_t , a_t)$, firstly a_t is introduced in the environment and the agent continues to receive the successor state $s_t{+}1$. At the very same time, the agent receives a $r_t \in R$ reward obtained from the reward function: $r_t = r(s_t , a_t , s_{t+1})$. The agent seeks to optimize the return $R_t = \sum^{\infty}_{k=0} \gamma^k r_{t+k}$ because of the factor of discount $\gamma \in [0, 1)$ by enhancing the policies effectiveness to $\pi*$. Our use of the actor–critic methodology with the student-t policy and regular temporal difference training at this stage. In this case, the function of value is also extended to include the expected value of the return from the present state: $V(s) = \mathrm{E}[R_t \mid s_t]$. The value function is used in such algorithm for the ds (dimensional state) space and the da (dimensional action) space $V(s) \in$ R and policy's three conditions are: location $\mu(s) \in \mathrm{R}^{x}{}_{y}$, scale $\sigma(s) \in \mathrm{R}^{x}{}_{y+}$ and degrees of freedom $\nu \in \mathrm{R}_{+}$, using function approximations, they must be approximated.

A. RL Architecture

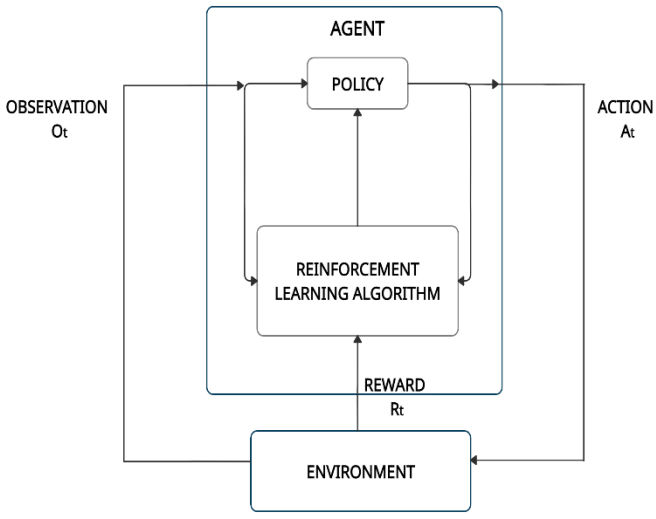

Fig.1. Reinforcement learning framework: an agent communicates with an environment by performing an action sampled from a policy over a current state; the environment then returns the next state based on dynamics and the reward to be maximized.

B. Proposed Architecture

We have built the architecture by incorporating the Gait analysis in two layers of standard format: Planning of Gait and Controlling of Gait.

Planning of Gait:

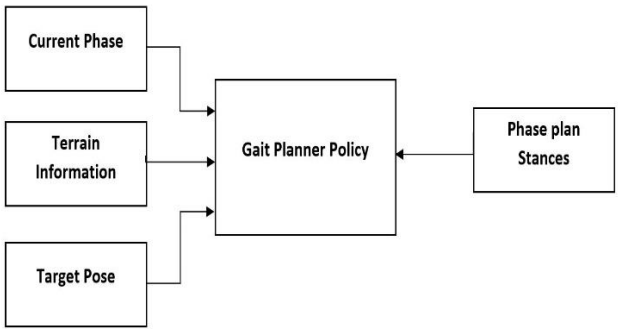

Fig.2. Planning of Gait: represents as a localized terrain-aware planner that generates a finite succession of support periods, i.e., a phase plan, using both exteroceptive and proprioceptive measures.

The Planning of Gait generates the aforementioned step plan by sequentially querying the planning policy $\pi_{\theta m}$. When proposing phase transitions, we formulate the Markov Decision Process in order to train $\pi_{\theta m}$ using RL, with the aim of ensuring that resulting policies learn to respect the robot's Kino dynamic properties and limits and also touch constraints. We can train $\pi_{\theta m}$ to gather a distribution on top of phase transitions that is not only feasible but also maximizes locomotion efficiency using the MDP that results.

Controlling of Gait:

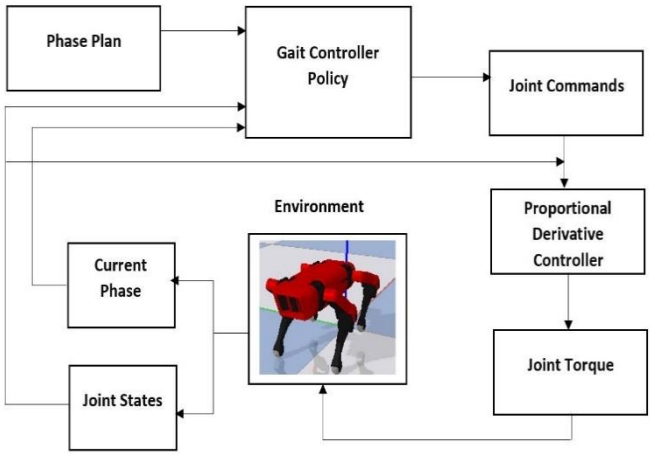

Fig. 3. Controlling of Gait: represents as the hybrid movement of planning system and controller that outputs joint location references using only proprioceptive sensing and the aforementioned step plan. Finally, a combined PD controller (with null goal joint velocity) computes joint torques using these joint position comparisons.

C. Policy Description

While learning from scratch may remove the need for human expertise and, in some cases, improve efficiency, having control over the learned policies is critical for robotic applications. For example, we might want to specify gait information. As a result, we split the locomotion controller into two components: an open loop component that enables the user provide the reference trajectories, and a response component that changes the leg poses based on the analysis.

$$a(t, o) = a^*(t) + \pi(o)$$

where $\pi(o)$ is the feedback component and $a^*(t)$ is the open loop component, which is normally a periodic signal. Users can quickly communicate their optimal gait using an open loop stimulus, and learning will take care of the rest, such as core stability, which is time-consuming to design individually. This hybrid policy is a broad formulation that provides users with a wide range of control options. It can be modified at any time, from completely user-specified to completely learned from scratch. We can set both the lower and upper limits of $\pi(o)$ to zero if we choose to use a user-specified policy. Set $a^*(t) = 0$ and give the feedback component $\pi(o)$ a broad output range if we want a policy that is learned from scratch. We can monitor how much user control is applied to the device by varying the open loop signal and the output bound of the feedback variable. We showed two examples in this paper: learning to gallop from scratch and learning to trot with a user-provided guide. We use a neural network to represent the feedback element π and Proximal Policy Optimization to solve the POMDP above.

D. Reward Function

Reinforcement learning optimizes a policy $\pi : O \rightarrow A$ that significantly increases the anticipated returns (accumulated rewards) R. A partial observation $o \in O$, rather than a complete state in $s \in S$, is observed at each control level.

We create a reward mechanism that motivates the agent to learn habits such as hitting the target position as quickly as possible, approaching the destination as fast as possible, reducing kinetic effort throughout phase transitions by avoiding long stances phases. The overall R_M i.e. reward function is composed of multiplicative and additive expressions as follows:

$$R_M (S_M,t, S_M,t+1) = r_m \cdot r^2_h \cdot r_k - r_c$$

where r_m awards the agent with the aim of moving the baseline foot hold position near to the target and inflict a penalty on the agent for driving it away, r_h penalizes the robot for not approaching the desired position, r_k inflict a penalty on for driving the gap between the marginal footholds under the shoulder and r_c inflict a penalty on for not raising a foot on the top of several steps, as a result, exploration is encouraged and such policies are kept from being trapped into the maximum of maintaining a constant stance.

In addition, we'd like to call attention to a few characteristics of the multiplicative word in the reward function above. This term produces a sanction that is low when r_p is less, i.e. at the start of session and high when r_p is more, i.e. at the end of the session, results in an automatic ascend of the total multiplication terms. We discovered that using these multiplicative rewards produces beneficial gradients across all training iterations because their attributes are never too big to stifle experimentation and really not that small to have no impact.

E. *Algorithm (PPO)*

PPO is a policy-based algorithm. PPO can be used in either discrete or continuous action space settings. Because of its easy usability and high performance, PPO has now become OpenAI's default reinforcement learning algorithm. To update the policy by maximizing the PPO objective is given by:

$$\theta_{k+1} = \arg max_{\theta} \frac{1}{|D_k|T} \sum_{\tau \epsilon D_k} \sum_{t=0}^{T} min\left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta}})\right)$$

Fit Value Function is given by:

$$\emptyset_{k+1} = \arg max_{\emptyset} \frac{1}{|D_k|T} \sum_{\tau \epsilon D_k} \sum_{t=0}^{T} (V_{\emptyset}(s_t) - R_t)^2$$

IV. Result

Following that, we created a collection of terrain based scenarios for priming and evaluating the controlling Gait and planning Gait agent's in order to evaluate our approach. The first and simplest scenario involves an infinite flat surface known as Flat-World, which we infer to create an output and behavior baseline. The terrain scenario is Temple-Ascent, which is a hybrid terrain that includes both flat and non-flat terrain areas. The RaiSim multi-body physics engine was used to create the MDP environment of the controlling Gait. All RL algorithms were implemented using the TensorFlow3.

A. *Simulations of quadruped in complex terrains*

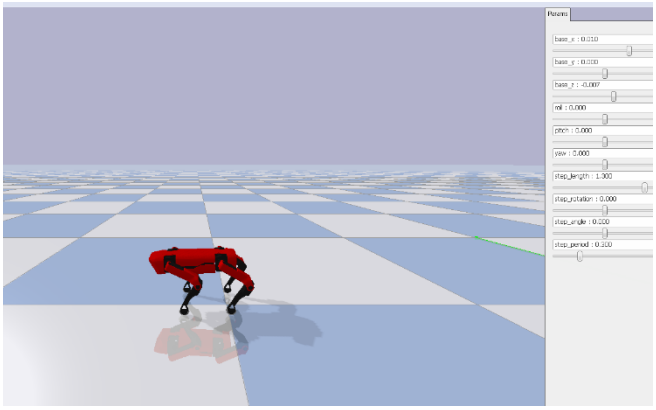

Fig. 4. Plane flat terrain: Basic approach towards simulation

Fig. 5. Hills non-flat terrain: Outdoor non-flat grass surface

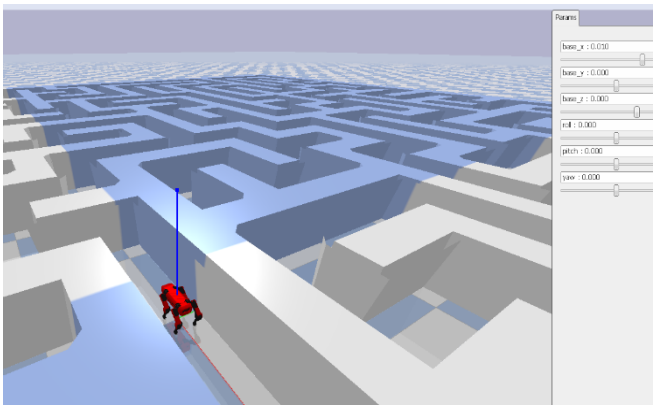

Fig. 6. Maze flat terrain: Obstacles based approach for movement

Fig. 7. Mountain non-flat terrain: Outdoor extreme complex non-flat surface

B. *Graph of cumulative reward function*

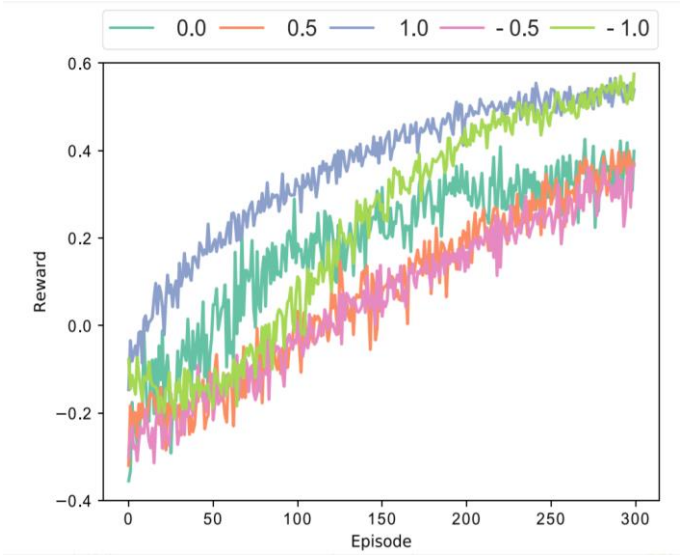

Fig. 8. Reward: In order to maximize reward, the agent must walk with respect to the set of episodes

C. Graph of lateral body posture & torqure

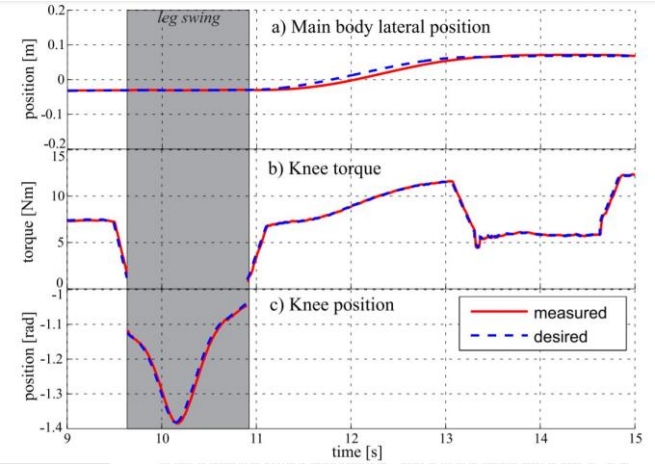

Fig. 9. Position and torque measured with respect to time

CONCLUSION

We've shown that RL can be used to teach robots agile locomotion on their own. We have demonstrated a complete framework that uses deep reinforcement learning in virtual environments and can learn from scratch or allow the user to direct the learning process. We successfully deployed the controllers trained in simulation on real robots using an appropriate physical model and reliable controllers. We were able to build two agile locomotion gaits for a quadruped robot using this system: trotting and galloping. Learning transferable locomotion policies is the subject of our paper. We used a simple incentive feature and a simple setting for this: optimizing running speed on a flat surface. To navigate the dynamic physical world in real-world situations, robots must see the environment, change their speed, and turn quickly. This suggests two promising directions for future research. First, we want to learn locomotion policies that can alter running speed and direction dynamically. Second, by incorporating vision as part of sensory feedback, this work could be extended to manage complex terrain structures.

REFERENCES

[1] M. Hutter et al., "ANYmal-a highly mobile and dynamic quadrupedal robot," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2016, pp. 38–44.

[2] S. Kuindersma et al., "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," Auton. Robots, vol. 40, pp. 429–455, 2016.

[3] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," IEEE Robot. Autom. Lett., vol. 3, no. 3, pp. 1560–1567, Jul. 2018.

[4] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in Proc. IEEE Int. Conf. Robot. Autom., 2018, pp. 1–8.

[5] D. Belter, P. Abcki, and P. Skrzypczyski, "Adaptive motion planning for autonomous rough terrain traversal with a walking robot," J. Field Robot., vol. 33, pp. 337–370, 2016.

[6] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," Int. J. Robot. Res., vol. 30, pp. 236–258, 2011.

[7] S. Tonneau, N. Mansard, C. Park, D. Manocha, F. Multon, and J. Pettré, "A reachability-based planner for sequences of acyclic contacts in cluttering environment," in Robotics Research. Berlin, Germany: Springer, 2018, pp. 287–303.

[8] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, S. Levine, Learning agile robotic locomotion skills by imitating animals. arXiv:2004.00784 [cs.RO] (2 April 2020).

[9] S. Ha, P. Xu, Z. Tan, S. Levine, J. Tan, Learning to walk in the real world with minimal human effort. arXiv:2002.08550 [cs.RO] (20 February 2020).

[10] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, V. Sindhwani, Data efficient reinforcement learning for legged robots, in Conference on Robot Learning (PMLR, 2019).

[11] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, S. Levine, Learning to walk via deep reinforcement learning (Robotics: Science and Systems, 2019).

[12] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, M. van de Panne, Learning locomotion skills for Cassie: Iterative design and sim-to-real, in Conference on Robot Learning (PMLR, 2019).

[13] J. Lee, J. Hwangbo, M. Hutter, Robust recovery controller for a quadrupedal robot using deep reinforcement learning. arXiv:1901.07517 [cs.RO] (22 January 2019).

[14] R. Wang, J. Lehman, J. Clune, K. O. Stanley, Paired open-ended trailblazer (POET): Endlessly generating increasingly complex and diverse learning environments and their solutions. arXiv:1901.01753 [cs.NE] (7 January 2019).

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

[16] C. D. Bellicoso, F. Jenelten, C. Gehring, M. Hutter, Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots. IEEE Robot. Autom. Lett. 3, 2261–2268 (2018).

[17] X. B. Peng, M. Andrychowicz, W. Zaremba, P. Abbeel, Sim-to-real transfer of robotic control with dynamics randomization, in IEEE International Conference on Robotics and Automation (ICRA) (IEEE, 2018).

[18] R. Hartley, J. Mangelson, L. Gan, M. G. Jadidi, J. M. Walls, R. M. Eustice, J. W. Grizzle, Legged robot state-estimation through combined forward kinematic and preintegrated contact factors, in 2018 IEEE International Conference on Robotics and Automation (ICRA) (IEEE, 2018), pp. 1–8.

[19] 7. M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, C. Semini, Probabilistic contact estimation and impact detection forstate estimation of quadruped robots. IEEE Robot. Autom. Lett. 2, 1023–1030 (2017).

[20] C. Gehring, C. D. Bellicoso, S. Coros, M. Bloesch, P. Fankhauser, M. Hutter, R. Siegwart, Dynamic trotting on slopes for quadrupedal robots, in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE, 2015), pp. 5129–5135.