Received January 8, 2021, accepted January 24, 2021, date of publication January 28, 2021, date of current version February 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3055241

Finite State Machine-Based Motion-Free Learning of Biped Walking

GYOO-CHUL KANG[1,2] AND YOONSANG LEE[2]
[1]CLO Virtual Fashion LLC., Seoul 06039, South Korea
[2]Department of Computer Science, Hanyang University, Seoul 04763, South Korea

Corresponding author: Yoonsang Lee (yoonsanglee@hanyang.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant NRF-2019R1C1C1006778 and Grant NRF-2019R1A4A1029800, and in part by the Research Fund of Hanyang University under Grant HY-2018.

ABSTRACT Recently, deep reinforcement learning (DRL) is commonly used to create controllers for physically simulated characters. Among DRL-based approaches, imitation learning for character control using motion capture clips as tracking references has shown successful results in controlling various motor skills with natural movement. However, the output motion tends to be constrained close to the reference motion, and thus the learning of various styles of motion requires many motion clips. In this paper, we present a DRL method for learning a finite state machine (FSM) based policy in a motion-free manner (without the use of any motion data), which controls a simulated character to produce a gait as specified by the desired gait parameters. The control policy learns to output the target pose for each FSM state and transition timing between states, based on the character state at the beginning of each step and the user-specified gait parameters, such as the desired step length or maximum swing foot height. The combination of FSM-based policy learning and simple linear balance feedback embedded in the base controller has a positive synergistic effect on the performance of the learned policy. The learned policy allows the simulated character to walk as instructed by the continuously changing the gait parameters while responding to external perturbations. We demonstrate the effectiveness of our approach through interactive control, external push, comparison, and ablation studies.

INDEX TERMS Character control, deep reinforcement learning, motion-free learning, locomotion control, physically based animation.

I. INTRODUCTION

Creating natural and responsive motion is one of the most important factors in bringing realism to virtual characters. Physics simulation is one of the effective ways to achieve this goal, as it always produces physically correct movements interacting with the environment. However, to make a physically simulated character perform desired actions while maintaining balance, we need a control algorithm that computes the control signals, such as joint torques, at every moment based on the state of the character and environment, as well as the intended goal.

Designing such a control algorithm that reproduces human locomotion skills has been one of the challenging topics in computer graphics. Various approaches have been proposed for locomotion control, some of which use reference motions for naturalness of the generated motion [1], [2], and some do not require reference motions and use other components such as finite state machines [3] or simplified dynamics models [4], [5] for generality of the controller.

In recent years, deep reinforcement learning (DRL) has received significant attention in the development of control algorithms. Formulating the DRL problem as finding a policy for tracking the reference motions (a.k.a. imitation learning) has been proven to be a powerful way to control a character to perform various human skills with natural movement [6]–[8]. Imitation learning-based approaches, however, tend to generate the motions that are limited to be near the reference motions. It is also not clear how to apply these approaches to extinct or difficult-to-capture animals.

The associate editor coordinating the review of this manuscript and approving it for publication was Wonhee Kim.

In motion-free learning that does not require reference motions, resulting motions are affected primarily by the design of the reward function, not by the specific style of a given motion. However, motion-free learning is difficult because it is not clear how to apply acceleration techniques such as the initial state distribution [6] and the reward design can become tricky. That is why the learning without motion data often results in controllers generating unrealistic and awkward gaits [9]. This low-quality motion problem can be effectively tackled using a curriculum learning method that allows for a high energy penalty [10], or by using realistic torque limits or the metabolic energy function [11], [12]. These approaches also help the control policy to avoid falling into the local minima during the learning process. For these purposes, we adopt another type of component, a finite state machine (FSM) based controller.

In this paper, we present a motion-free DRL method for learning a controller for a physically simulated character that produces a gait as specified by the desired gait parameters, without the use of any motion data. The control policy representation is based on a FSM-based control algorithm similar to SIMBICON [3]. Our policy learns to output the target pose to be tracked for each FSM state and transition timing between states based on the character state at the beginning of each step and the given gait parameters. The learning process is facilitated by a linear feedback law embedded in the FSM-based controller. The combination of FSM-based policy learning and simple linear feedback has a positive synergistic effect. Using this combination results in a controller that outperforms controllers using each item alone. For example, our policy learned with the feedback law responds better to external pushes than a controller using only the feedback law by increasing the step frequency to take more steps. The learned controller can generate a simulated movement according to the specified gait parameters such as the desired step length or maximum swing foot height.

We use a number of technical components to facilitate effective learning of the FSM-based control policy. 1) The simulation time interval between action queries is different each time in our FSM-based DRL formulation, unlike previous DRL-based controllers that use a fixed simulation duration (e.g. $1/30$ s) as the interval. We use a simple reward scaling method to address the problem in which the reward for a certain length of simulation can be evaluated differently depending on the number of RL time steps involved, which can occur in our formulation. 2) We change the FSM-based control algorithm in terms of the stance hip torque and swing-up state duration to support a more natural and diverse gait. 3) We introduce the FSM-based action range, which provides an effective way to learn a policy for a stable gait while avoiding the local minima, despite the absence of a reference motion. We demonstrate the performance of our controller and the effectiveness of the design choices through interactive control, external push, comparison, and ablation studies.

II. RELATED WORK

Designing locomotion control algorithms for physically simulated characters that generate natural human movements has been a notorious challenge for decades. The balancing mechanism, which is a mechanism for a simulated character to maintain balance even under unexpected disturbances, is the key for the design of a control algorithm. One of the earlier approaches is to simplify the problem by representing a locomotion cycle as a finite set of states and transitions between them, along with manually designed balance feedback laws [3], [13], [14]. This approach has shown that the combination of a FSM and manual feedback laws with hand-tuned parameters can be an effective way to produce robust locomotion, despite showing a somewhat marching-like, stereotyped gait. Based on these studies, Wang *et al.* proposed to optimize the parameters in the FSM states and feedback laws with the objectives for naturalness and controllability of the generated motion using stochastic optimization [15], [16]. They showed that the stochastic optimization can be an effective tool to optimize a FSM-based controller to generate energy-efficient and natural movements, without the use of motion capture data. Simplified dynamics models such as an inverted pendulum or centroidal dynamics [17] have been used in the design of locomotion control algorithms to alleviate the complexity of full-body dynamics [4], [5], [18], [19]. The use of motion capture data has been another popular approach to design a controller that generates human-like movements [1], [2], [20]. Although our control policy is also based on a FSM, because it uses DRL rather than stochastic optimization, a single controller instance can cope with various situations such as various desired gait parameters or external perturbations.

DRL has recently shown impressive progress in the development of locomotion controllers. A popular approach is to use DRL to find a policy that can reproduce reference motions, which is an effective way to achieve the quality of generated motions and the diversity of the actions performed. Deep Q-learning [21] has been used to learn how to schedule control fragments for short motion segments to perform various tasks such as running, skateboarding, walking on a ball, and dribbling a basketball [20], [22]. Policy gradient methods, such as proximal policy optimization (PPO) [23], have been widely used to find a tracking policy on a continuous action space. It has been successfully adopted to learn PD target poses to perform a variety of actions mimicking the given reference motions while following the specified task objectives [6]. By sampling the initial state of each episode from the reference motion to preferentially explore the states around the desired motion, this approach not only provides naturalness to a simulated motion it also allows efficient learning of the tracking policy. Learning a control policy for a large set of unstructured motion capture data can be facilitated by motion matching [8] or RNN [7] based motion generators. In order to train a policy for more challenging tasks, adaptive sampling [7], [24] or curriculum learning [10], [12] in task or environmental parameter spaces have been proposed. Without using motion data, a controller learned by DRL has often exhibited a non-human-like gait [9]. To improve the quality of generated motion without using a reference motion, a combination of the mirror symmetry loss and curriculum learning that allows for a high-energy penalty was proposed to learn a policy that produces human-like, low-energy locomotion [10]. Curriculum learning in the task parameter space with an adaptive curriculum can result in a robust control policy for stepping-stone environments without the use of motion data [12]. Our DRL framework also does not use reference motion data to train a locomotion controller. We employ a FSM-based control algorithm with a linear balance feedback law to compensate for the absence of reference motion data. Our framework trains a successful locomotion control policy that can exhibit a natural gait for various gait parameters.

There are two studies close to ours in that FSM and reinforcement learning are used together, with the following differences. Coros *et al.* proposed a policy that learns to choose one of the predefined FSM controllers with different gaits on a discrete action space [25]. The policy is learned through value iteration over the trusted states that are carefully extended. Our policy is learned over a more expressive continuous action space through DRL and does not need a set of manually tuned controllers. Peng *et al.* proposed a policy more similar to ours in that it outputs the target angles directly to create a terrain adaptive gait using terrain information [26]. However, it works in a 2D environment and requires a number of FSM controllers to create specialized actors in the mixture of actor-critic experts structure. Ours works in a 3D environment without a set of pre-tuned FSM controllers, and creates a gait by using user-specified gait parameters.

III. FSM-BASED CONTROL

Our control policy representation is based on a FSM-based controller that resembles SIMBICON [3]. The FSM has four states for a single locomotion cycle, where each state has the target angles of all actuated joints for PD control (Figure 1). States 0 and 1 define the target poses for the right swing phase, and states 2 and 3 define those for the left swing phase. The character moves its swing leg forward during states 0 and 2 (swing-up states), and lowers the swing leg during states 1 and 3 (swing-down states) to make contact with the ground. Note that the term "left swing phase" does not refer to the phase where the left foot is in the air and the right foot is in contact with the ground, but refers to the phase in which the target poses are set to guide the swing of the left leg. The target swing hip angles in all states are modified in both the sagittal and coronal planes using a simple feedback law:

$$\theta_d = \theta_{d0} + c_d d + c_v v, \tag{1}$$

where θ_d is the modified target angle, θ_{d0} is the target angle defined in the states, d is the horizontal distance from the

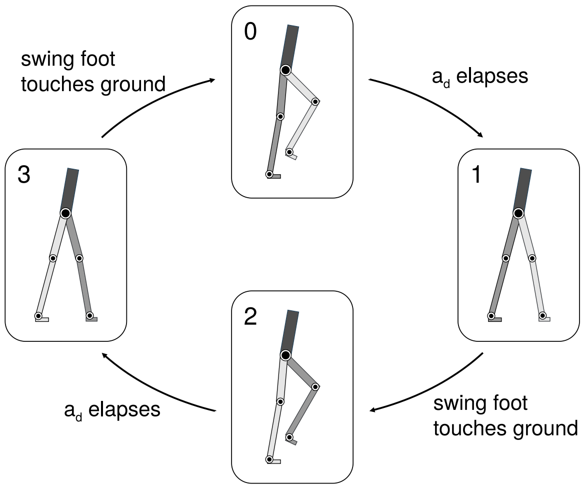

FIGURE 1. The FSM has four states; states 0 and 1 for the right swing phase, states 2 and 3 for left. Each state has a target pose for PD control. a_d is the swing-up state duration that is determined by our control policy. Note that, although each target pose is shown as a planar biped only with the root link and two legs for simplicity, each FSM state consists of the target angles of all actuated DOFs.

stance ankle to the pelvis, v is the horizontal speed of the pelvis, and c_d and c_v are the balance feedback gains. The joint torques are then computed from the modified target pose using the PD control.

Note that, in principle, our control policy can be applied to a general biped character with two legs and a pelvic part as the root link, regardless of the structure of the remaining joints. It can be a 3D character where the feedback law is applied to a 3-DOF swing hip joint both on the sagittal and coronal plane, or even can be a 2D character where it is applied to a 1-DOF swing hip joint on the sagittal plane. We use 3D characters for our results, and in this case, c_d and c_v actually consist of four parameters in the sagittal and coronal planes (c_d^{sag}, c_v^{sag}, c_d^{cor}, c_v^{cor}). Remaining joints, such as knees, ankles, and upper-body joints (if exist) are controlled by the target angles in each FSM state using PD control.

In order to support more natural and diverse gaits, we modify the SIMBICON algorithm in terms of the swing-up state duration and stance hip torque computation.

Swing-up state duration. In SIMBICON, a transition between the states occurs when the swing foot hits the ground (state 1 to 2 and 3 to 0) or a fixed time elapses (state 0 to 1 and 2 to 3). In our controller, the latter transition condition is set as a variable a_d, which is determined by the control policy as a part of the actions. This enables the generation of more diverse gaits because the duration of the swing-up states considerably affects the characteristics of locomotion behavior. The learned control policy chooses the best duration for the swing-up states for the given gait parameters.

Stance hip torque computation. We compute the stance hip torque from a target pose just like other joints, without applying a virtual force to the root link. In SIMBICON, this is calculated as the negated sum of the swing hip and root link torque, which are computed using virtual PD control, to ensure that only internal torques are used. This has the advantage of reducing the number of hand-tuned parameters, and thus it can be a simple way to find the stance hip torque that generates a stable gait. However, we find that this method tends to result in a less natural, marching-like gait when applied to our control policy. By leaving the computation of the stance hip torque as an area for DRL to learn, the learned policy shows a more natural and smoother gait. Note that our controller does not use SIMBICON's virtual PD control on the root link, and thus the character is actuated only by internal joint torques.

IV. REINFORCEMENT LEARNING

We use reinforcement learning (RL) to control the character equipped with the FSM-based control algorithm. In RL, an agent observes an environment state $\mathbf{s}_t$ and chooses an action $\mathbf{a}_t$ to take at every RL time step. After conducting $\mathbf{a}_t$, the agent receives a reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$ from the environment. The goal of RL is to find the policy $\pi_\theta(\mathbf{a} \mid \mathbf{s})$ that maximizes the cumulative reward:

$$J(\theta) = E\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right], \tag{2}$$

where θ is the parameter of the policy and $\gamma \in (0, 1)$ is the discount factor for the rewards. In DRL, a policy π is a deep neural network with the network parameter θ. We use the proximal policy optimization (PPO) [23] to find the best network parameter θ.

We use a single walking step as a RL time step to exploit the discrete nature of the FSM-based control algorithm where the FSM transitions and target pose updates occur mainly based on steps. Thus, one RL time step contains two state transitions in the FSM. Another alternative is to use each FSM state as a RL time step, which would degrade the learning performance due to the delayed reward because the reward for the given gait parameters, such as the desired step length, must be calculated for every two RL time steps. Another problem with this alternative is that one of the actions, the swing-up state duration, only makes sense every two RL time steps, because it is only used in the swing-up states.

A. STATE

The state $\mathbf{s}$ is composed of the dynamic state $\mathbf{s}_d$ of the simulated character, the user-specified gait parameter $\mathbf{s}_p$, and the binary indicator s_s showing which leg is about to swing. Note that $\mathbf{s}$ is computed at each RL time step, that is, at the beginning of each walking step. With the support of the balance feedback law (Equation 1), the learned policy can find an action that creates a stable gait with only the state at the beginning of each step.

The dynamic state $\mathbf{s}_d$ includes the generalized position $\mathbf{q}$ and velocity $\dot{\mathbf{q}}$ of the root joint and the joints of both legs. Additional joint states may be included depending on the simulated character and environment. For example, if the simulated character has upper-body parts, it may contain some of the upper-body joint states. The end-effector positions $\mathbf{e}$ with respect to the root frame are included to provide more information about the current pose. The desired gait parameter $\mathbf{s}_p$ is the vector of user-specified desired values for step length l_d, step duration d_d, and maximum swing foot height during a step h_d (Figure 2).

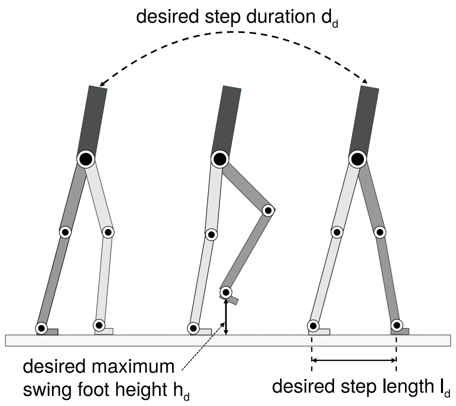
desired step duration d_d
desired maximum swing foot height h_d
desired step length l_d

FIGURE 2. Schematic diagram of the desired gait parameter $\mathbf{s}_p$.

B. ACTION

The action $\mathbf{a}$ consists mainly of the two sets of target angles $\mathbf{a}_p$ for the joints of both legs and other optional joints. It is responsible for creating one walking step. Each set of target angles corresponds to the swing-up and swing-down state of the FSM. The target angles of the joints not included in the action $\mathbf{a}$ are set to constants showing a fixed posture or periodic movement. As described in Section III, the swing-up state duration a_d is also included in the action $\mathbf{a}$.

FSM-based action range. When people walk, they repeatedly swing their swing leg up and down by adjusting the range of motion of the hip and knee joints on the sagittal plane, while their movements on other anatomical planes and the motion of other joints remain within a certain range. This means that the typical target angles for the swing-up and swing-down states could differ for specific joints. Based on this idea, we set the range of the target joint angle $\mathbf{a}_p$ that depends on the FSM state (see Table 1 for an example). In this range, the target ranges for the hip and knee on the sagittal plane vary depending on the FSM state, whereas the ranges for the other DOFs are fixed. For example, the target swing hip angle is set to have a positive value for the swing-up state and a negative value for the swing-down state, which results in flexion and extension of the swing hip, respectively. This action range helps prevent the policy from being stuck to the local minima during the learning process. Without the action range, the policy only learns to stand still while slightly moving body parts.

C. REWARD

The reward r is composed of five terms that allow the character to move forward while maintaining an upright posture

TABLE 1. **An examplar FSM-based action range for the *Atlas* character. The character is in a standing position when the angle of all joints is 0°. The sets of swing and stance leg joints are switched to the left or right leg based on the FSM state.**

Joint	Action range	
	swing-up (state 0, 2)	swing-down (state 1, 3)
swing hip flexion(+) / extension(-)	0° – 90°	-30° – 0°
swing knee flexion(-) / extension(+)	-100° – -70°	-20° – 0°
stance hip flexion(+) / extension(-)	-30° – 0°	
stance knee flexion(-) / extension(+)	-20° – 0°	
swing & stance hip abduction(+) / adduction(-)	-10° – 10°	
swing & stance hip ext.(+) / int. rotation(-)	-30° – 30°	
swing & stance ankle dorsi(+) / plantarflexion(-)	-20° – 20°	
swing & stance ankle inversion(+) / eversion(-)	-5° – 5°	
back flexion(+) / extension(-)	-30° – 30°	

and satisfying the desired user-specified gait parameters with minimum effort.

The first term E_{param} penalizes the deviation in the gait parameters of the current step from the desired gait parameters:

$$E_{\text{param}} = -\left\|\mathbf{s}_p - \mathbf{m}_p\right\|_1, \tag{3}$$

where $\mathbf{s}_p$ is the desired gait parameter vector and $\mathbf{m}_p$ is the gait parameter vector measured from the simulation of the current step.

The second term E_{up} encourages the character to maintain its upper-body upright:

$$E_{\text{up}} = \hat{\mathbf{v}}_{\text{root}} \cdot \hat{\mathbf{e}}_{\text{vertical}} - 1, \tag{4}$$

where $\hat{\mathbf{v}}_{\text{root}}$ is the unit vector representing the current vertical direction of the root link and $\hat{\mathbf{e}}_{\text{vertical}}$ is the vertical axis vector of the world frame.

The next term E_{fwd} penalizes the deviation from the forward direction:

$$E_{\text{fwd}} = -\left|c_{\text{lateral}}\right|, \tag{5}$$

where c_{lateral} is the character's pelvis position in the lateral axis.

The effort term E_{eff} penalizes excessive joint torques during the current step:

$$E_{\text{eff}} = -\frac{1}{T}\int_T \sum_i \|\tau_i(t)\|\, dt, \tag{6}$$

where i is the joint index, $\tau_i(t)$ is the torque for each joint at time t, and T is the time duration for the current step.

The total reward r is defined as follows:

$$r = (w_1\, E_{\text{param}} + w_2\, E_{\text{up}} + w_3\, E_{\text{fwd}} + w_4\, E_{\text{eff}} + E_{\text{alive}}) \cdot d_d, \tag{7}$$

where E_{alive} is the alive bonus for an unterminated episode at the current step and w_{1-4} are the weight for each term.

Reward scaling. In our formulation, the RL time step is based on a step, not on a fixed-length time slot. One difficulty with this is that the rewards from steps of different duration are not evaluated equally in terms of the same simulation time. Specifically, an agent taking more number of shorter duration steps gains more reward than another agent with less number of longer duration steps in episodes of the same length. It means that sufficient rewards are given for less-optimized shorter duration steps, compared to longer steps, due to the higher number of steps. Thus, the policy generates a less-optimized gait when instructed to generate shorter steps.

The weighted sum of the rewards in Equation 7 is scaled by the desired step duration d_d to address this problem. If the reward is not scaled by d_d, the cumulative reward for an episode where d_d is given small will be overrated. For example, suppose that in two 5 second long episodes with d_d of 0.5 seconds for the episode e_1 and 1 second for the episode e_2, all walking steps fully satisfy the given $\mathbf{s}_p$ and the weighted sum of the rewards for each step is 1. Scaling by d_d, the reward for e_1 is $5 = 1 \times 0.5 \times (5/0.5)$ and that for e_2 is $5 = 1 \times 1 \times (5/1)$ under the assumption that the discount factor is 1. Without scaling, the reward for e_1 and e_2 is $10 = 1 \times (5/0.5)$ and e_2 is $5 = 1 \times (5/1)$, respectively. Therefore, if d_d is small, the agent can obtain the same level of reward even if the weighted sum of the rewards is smaller than when d_d is large. As a result, the policy generates less-optimized motions with respect to the reward terms in such cases. Scaling the reward by d_d allows us to equally evaluate the "quality" of shorter and longer step duration steps in the same length of simulation time.

An alternative is scaling the reward by the current step duration measured from the simulation instead of its desired value d_d. This alternative, however, fails to make the policy learn how to walk, probably because the learning process focuses on creating long-duration steps in the short term.

D. NEURAL NETWORK TRAINING

The policy is represented by a deep neural network with three fully-connected layers, where each hidden layer has 64 units. We use the tanh activation for all hidden layers. The output action is scaled to meet the FSM-based action range (Table 1) based on the current state of the FSM. The value function network has a structure identical to the policy network, except for the output dimension of 1.

The PPO algorithm collects experience tuples to update the policy and value network, from a number of episodes. At the beginning of each episode, the character is initialized to a standing pose with both arms down. The desired parameter $\mathbf{s}_p$ is uniformly sampled from the predefined range at the beginning of the episode and after 3 seconds, allowing the policy to learn how to satisfy the variety of gait parameters and transitions between them. The range of each desired parameter should be reasonable, allowing it to be achieved by the character based on a learned policy. For example, walking while satisfying both a desired short step duration (e.g. 0.1 s) and a desired long step length (e.g. 1.2 m) can be physically difficult. Sampling such an unreasonable point in the parameter space can only lead to waisted learning time. We carefully set the range of parameters to $[0, 0.5]$ for d_d, $[d_d/3, d_d/3 + 0.2]$ for l_d, and $[d_d/4, d_d/4 + 0.15]$ for h_d to avoid such cases. The minimum and maximum values of d_d, l_d, and h_d are used to normalize the gait parameter state $\mathbf{s}_p$ for network training. The dynamic state $\mathbf{s}_d$ is normalized using the joint angle and velocity limits of the simulated character. The episode is terminated if the character is detected as falling by checking the height of the pelvis, to discourage undesirable behaviors and collect more relevant tuples as discussed in [6]. The maximum length of an episode is 6 seconds.

V. RESULTS

We use PyDart2 [27] for forward dynamics simulation with a simulation frequency of 900 Hz. At each simulation time step, the target swing hip angle in the queried action $\mathbf{a}$ is adjusted by the balance feedback law (Equation 1). The feedback gains c_d^{sag}, c_v^{sag}, c_d^{cor}, and c_v^{cor} are set to values that allow the simulated character to walk steadily with our implementation of SIMBICON. We found that the choice of feedback gains is not a significant factor for the robustness of our control policy if they are within a reasonable range. The joint torques are computed from the adjusted target pose using PD control at each simulation time step. Note that we use PD control instead of stable PD [28] which is currently very popular. The low-stiff nature of the FSM-based control algorithm allows us to use more computationally efficient naive PD control, as little effort is required to tune the PD gains and time step size so that the controller can generate sufficient torques to move the simulated character. Note that since our SIMBICON implementation does not have any part for adjusting the walking direction, the walking direction of the SIMBICON-controlled character may change in certain cases in the comparison results.

For DRL, we use an open-source implementation of the PPO algorithm [29]. We use the Adam optimizer [30] to update the network weights. The learning rates for the policy and value networks are 1×10^{-4} and a PPO clipping range of 0.2 and a discount factor of 0.99 are used. Learning a policy takes approximately 72 hours on a machine with a 16-core AMD Ryzen Threadripper 1950X CPU and Nvidia Geforce RTX 2080 Ti GPU, with simulation running in parallel on 32 threads.

A. LEARNING "STANDARD" POLICIES

The *Atlas* character is used in most of our simulation results. It has 33 DOFs including six unactuated DOFs with 28 links of 120 Kg in weight and 1.5 m in height. We use a publicly available specification of the Atlas robot without a head [31] and modify it to replace the inertia of the upper torso with the no-backpack version inside the specification file. $\mathbf{s}_d$ includes the states of the root, lower-body joints, and back joint which connects the root link and lower torso, and the position of five end-effectors (the feet, hands, and upper torso). $\mathbf{a}_p$ consists of the target angles for the lower-body joints and back joint. The target angles of the upper-body joints are fixed in an upright position, except for the shoulder joints, where the target angles are predefined for each FSM state to create a slight arm swing movement. The FSM-based action range for this chracter is listed in Table 1.

We refer to the policy learned using the components described in the previous sections as the *standard* policy. The balance feedback gains of $c_d^{sag} = 0.5$, $c_v^{sag} = 0.2$, $c_d^{cor} = 0.5$, and $c_v^{cor} = 0.2$ are used in the learning process. All the policies used to create the results are learned over 1050 iterations, and each iteration uses 16,384 experience tuples. If the evaluated cumulative rewards decrease before 1050 iterations, we use the network with the highest reward.

B. MANIPULATION OF DESIRED GAIT PARAMETERS

We verify the performance of the *standard* policy on varying the desired gait parameters $\mathbf{s}_p$. In the accompanying video, the desired and measured gait parameters are visualized through the length of the colored bars (Figure 3).

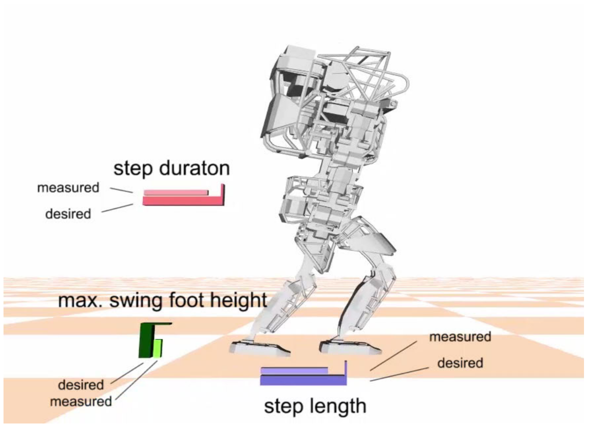

FIGURE 3. Colored bars visualize the desired (bars with a tip) and measured (bars without a tip) gait parameters. Measurements are taken from the previous step simulation.

To create continuous changes in $\mathbf{s}_p$, we use the *increase & decrease* scenario where each parameter in $\mathbf{s}_p$ is set to the median of its range for 3 seconds, and then gradually increases from the minimum to maximum and back again to the minimum, for approximately 20 seconds. Under this scenario, the policy allows the simulated character to walk in a variety of ways, from short and quick steps to long and slow steps. A user can also interactively change $\mathbf{s}_p$ to change the locomotion style of the simulated character. As shown in the accompanying video, the simulated character can walk steadily while changing its gait style as directed by the user.

C. EXTERNAL PUSHES

We investigate the robustness of the learned policy with two types of push tests.

We apply pushes of 500 N with a duration of 0.1 seconds at the torso from the front, rear, right, and left directions. A push is applied every ten steps when a transition from state 0 to 1 occurs. The *standard* policy allows the simulated character to continue walking while interactively changing the desired gait parameters with these unexpected external pushes.

We compare the push-recovery capability of our *standard* policy and our implementation of SIMBICON. Our SIMBICON implementation uses the same set of balance feedback gains as those used in the *standard* policy. The parameter $\mathbf{s}_p$ for the *standard* policy is given consistently to produce a gait that is similar to that of SIMBICON. The 600 N, 800 N, 1000 N, and 1200 N external pushes, which last 0.1 seconds each, are applied to the back of the character controlled by each controller every 10 to 15 seconds, providing sufficient time to restore the balance and stabilize the cycles.

Table 2 summarizes the comparison results. Our policy starts to fail from 1200 N, while SIMBICON starts to fail from 800 N with the identical feedback gains. For the 1200 N pushes, ours withstands one push and fails on the second push but SIMBICON fails on the first push. The measured rate of step frequency increase indicates that the ability to adjust step frequency contributes this robustness. For all magnitude pushing force, the step frequency increase rate of our policy is higher than that of SIMBICON, meaning that our policy increases the number of steps more than SIMBICON to avoid falling down. The reason is that our policy can directly increase the step frequency by selecting a smaller a_d, whereas SIMBICON only responds by flexing the swing hip joint further based on its balance feedback law and thus adjusts the step frequency indirectly. Another observation is that the step frequency increase rate increases as the pushing force increases for the both controllers. If the force is weak, the step frequency may rather decrease. However, it is the same that the increase rate increases as the force gets stronger. It can be seen that our controller shows a higher rate of increase overall and responds effectively to external force.

TABLE 2. Comparison of our *standard* policy and SIMBICON on the push tests. The step frequency increase rate is measured on the first push of each test, using the time to take five steps before and after the push. Note that there is no measurement for SIMBICON with 1200 N push because the character falls down right after the first push.

Controller	Number of pushes until the character falls (Step frequency increase rate after a push)			
	600 N	800 N	1000 N	1200 N
Ours	Not falls (-8.47%)	Not falls (3.52%)	Not falls (15.26%)	2 (25.0%)
SIMBICON	Not falls (-25.16%)	2 (-1.64%)	2 (11.51%)	1 (N/A)

This observation indicates that the balance feedback is only a part of the push-recovery capability of our policy. This implies that, despite the absence of external pushes during the learning process, our policy learns to find actions that help maintain balance by referring only to the character state at the beginning of each locomotion step.

D. BALANCE FEEDBACK

We first test the sets of feedback gains that are different from those used in the learning process of the *standard* policy in order to examine the resilience to changes in the feedback gains. Then we investigates the contribution of the balance feedback.

First, we apply the scaled balance feedback gains of 0.4, 0.6, 0.8, 1.2, 1.4, and 1.6 times that used to learn our *standard* policy, to the simulated character that is controlled by the *standard* policy (Table 3). We use the *increase & decrease* scenario to test the robustness at various gaits. The character simulated with 0.4× feedback gains falls quickly (within about 5.3 s), but walks longer as the gain increases. It can continue walking without falling using the feedback gains scaled by factors of 1.2× and 1.4×. As shown in the accompanying video, both successful feedback gains lead to similar gaits. This means that our policy can successfully find a target pose that compensates for the excessive movement of the swing leg due to the large feedback gain, to create a stable gait and maintain balance.

TABLE 3. Resilience of our *standard* policy to the scaled balance feedback gains of 0.4, 0.6, 0.8, 1.2, 1.4, and 1.6 times that used to learn the policy under the *increase & decrease* scenario.

The amount of time until the character falls (sec)					
0.4x	0.6x	0.8x	1.2x	1.4x	1.6x
5.29	11.73	20.06	Not falls	Not falls	16.83

Second, we compare our *standard* policy and SIMBICON with the same set of the scaled feedback gains from 0.8× to 1.6× (Table 4). Again, $\mathbf{s}_p$ for the *standard* policy is set to produce a gait similar to that of SIMBICON. Although SIMBICON can balance a few more seconds (15.16 s) over our *standard* policy (11.46 s) for 0.8× feedback gains, the *standard* policy can maintain balance with a wider range (1.2×-1.6×) of feedback gains than SIMBICON (1.2×-1.4×) over the long run. Note that the *standard* policy is successful over a wider range of feedback gains than that in the first test because of the fixed $\mathbf{s}_p$.

TABLE 4. Comparison of our *standard* policy and SIMBICON on the scaled balance feedback gains.

Controller	The amount of time until the character falls (sec)			
	0.8x	1.2x	1.4x	1.6x
Ours	11.46	Not falls	Not falls	Not falls
SIMBICON	15.16	Not falls	Not falls	5.3

Lastly, we investigate the contribution of the balance feedback law for maintaining balance. In the RL formulation where the state is only updated at the beginning of each locomotion step, the ability of the policy to maintain balance is limited because it is difficult for the character to react to state changes in the middle of a step. To investigate this aspect, we train two policies: the *no-feedback* policy that is learned without the balance feedback (using 0 for the feedback gains) and the *no-feedback-swingup* policy that is learned with the balance feedback disabled only in the swing-up states. Specifically, in the *no-feedback-swingup* policy, the state changes during a step are expected to be addressed by the balance feedback in swing-down states.

The learning curves of the policies are shown in Figure 4, which shows the average return of 10 test episodes. Notably, the *no-feedback-swingup* policy learns faster than the *no-feedback* policy, meaning that using the balance feedback only in swing-down states helps the character quickly learn how to walk. However, the *no-feedback-swingup* policy achieves a lower return than the *standard* policy. As a result, as shown in the accompanying video, not using the balance feedback at all or using it only in swing-down states is not enough to learn a stable walking controller. Indeed, both of the policies cannot maintain balance for a long time, even if they learn more than twice the *standard* policy.

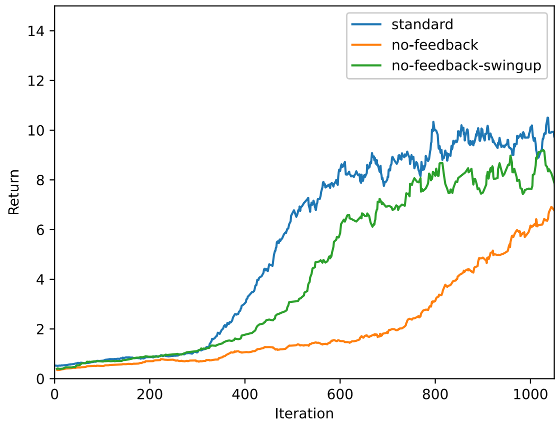

FIGURE 4. **Learning curves for the *standard*, *no-feedback*, and *no-feedback-swingup* policies.**

From the comparison with SIMBICON for external pushes and scaled feedback gains, we can see that a policy that takes only the character state at the beginning of a step can contribute to maintaining balance. The ablation studies for the balance feedback, however, shows that such a policy alone without the balance feedback has difficulty taking full responsibility for balancing. This means that FSM-based policy learning and simple linear balance feedback complement each other to produce a more robust locomotion controller.

E. ABLATION STUDIES

We perform ablation studies to investigate the effectiveness of our major design decisions.

Learning without the FSM-based action range results in the *no-action-range* policy that cannot make the simulated character walk. Nevertheless, Figure 5 shows that the *no-action-range* policy learns faster in the early stages than the *standard* policy. This is because learning to stand still without falling is easier than learning to balance while taking steps. Although the *no-action-range* policy quickly learns how to not fall down, it only achieves a fairly lower return (about 7) than the *standard* policy (over 10) because it does not satisfy the desired parameters $\mathbf{s}_p$ at all. As a result, the character just maintains a standing position by slightly moving its body parts (Figure 6, second row). Unlike the previous DRL-based controllers that use reference motions, which learn to perform the desired behavior effectively by preferentially exploring the states around the reference motion, our controller does not have any reference motion to track. Thus, it is easy to fall into the local minima without any additional effort, such as designing more sophisticated rewards. The FSM-based action range provides a simple and effective way to create a successful gait without falling into the local minima, by specifying the range of possible actions according to the nature of each FSM state.

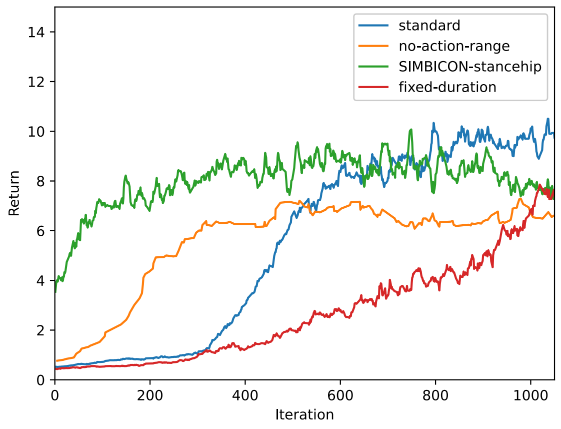

FIGURE 5. **Learning curves for the *standard*, *no-action-range*, *SIMBICON-stancehip*, and *fixed-duration* policies.**

The *SIMBICON-stancehip* policy is learned using SIMBICON-style stance hip torque calculation (see Section III) instead of computing it from the target angle output by the policy network. Interestingly, the return from the *SIMBICON-stancehip* policy shows a value of around 4 from the beginning, and achieves near-peak values much faster than the *standard* policy (Figure 5). This implies that the SIMBICON's stance hip torque computation is advantageous for quickly finding a way to walk without falling down, which is probably why it could be successfully applied to the original work that manually tunes the control parameters [3]. However, this strategy turns out to produce a less natural, marching-like gait when applied to our framework, and the maximum return is lower than that of the *standard* policy. The learned policy generates a SIMBICON-like gait that shows a quick swing down of the swing leg (Figure 6, third row). It is

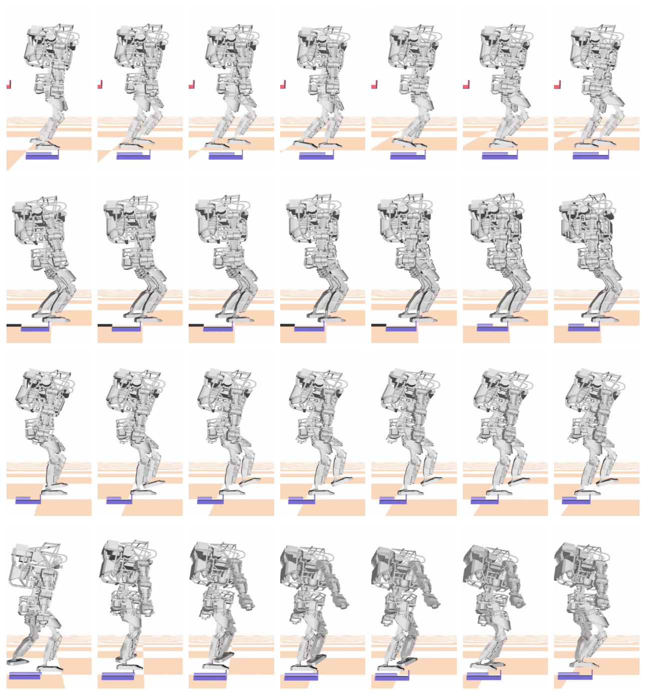

FIGURE 6. The simulated results of the *standard*, *no-action-range*, *SIMBICON-stancehip*, and *fixed-duration* policies (top to bottom). The images are captured at 0.133 second time intervals for each column.

interesting to note that the change in stance hip control leads to a change in swing-down movement. This result implies that the movement of the stance hip can be an important factor in the natural movement of the swing leg.

The *fixed-duration* policy is learned using a fixed duration of 0.3 seconds for the swing-up states, as in SIMBICON. This policy learns much slower than the *standard* policy or aforementioned two variants of the policy (Figure 5). The learned policy can make the character walk but with an exaggerated lateral movement of the upper-body and the simulated step duration is almost constant regardless of the desired step duration (Figure 6, fourth row). As described in Section III, the introduction of the swing-up state duration as an action allows the character to demonstrate various walking styles.

F. REWARD SCALING

To investigate the effect of reward scaling by d_d (Equation 7), we examine two policies: a policy learned with reward scaling using the simulated step duration instead of desired duration, and a policy learned without reward scaling.

It may seem more reasonable to use the simulated step duration to eliminate the difference in the reward evaluation based on the duration of a single step. However, the *reward-sim-duration* policy learned using the simulated step duration cannot make the character walk. This is probably because the learning process focuses on creating long-duration steps in the short term and, as a result, fails to learn how to balance, because the simulated duration multiplied by the reward dominates the other reward terms and the output action directly affects the simulated duration.

Without reward scaling, the *no-reward-scaling* policy cannot sufficiently learn to satisfy the small d_d (Figure 7). We test this policy under a scenario in which each item of $\mathbf{s}_p$ is given as a minimum, which means that the desired gait is walking with quick steps of a fairly short step duration. The *no-reward-scaling* policy does not sufficiently satisfy the given $\mathbf{s}_p$, whereas the *standard* policy does satisfy it and produces a stable and sustained gait.

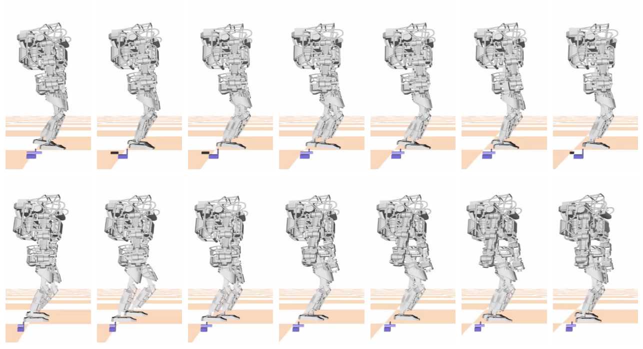

FIGURE 7. Standard (top), no-reward-scaling (bottom) policies with the minimum desired gait parameters. The images are captured at 0.133 second time intervals for each column.

Addtionally, we tested the number of steps for reward scaling in the early stage of the study. But we found that multiplying the number of steps guides the policy to always generate quick and short steps because it dominates the reward overwhelmingly over the others. The policy also failed to learn to walk with rewards divided by the number of steps as there was no incentive for long walks. Even if the agent walks one or two steps and falls early, it is well rewarded for achieving the desired parameters of that one or two steps.

G. MORE CHARACTERS

We train the *standard* policies for two more characters (Figure 8). The same balance feedback gains used for the *Atlas* character are also used for these two characters. The trained policies control these characters to walk stably.

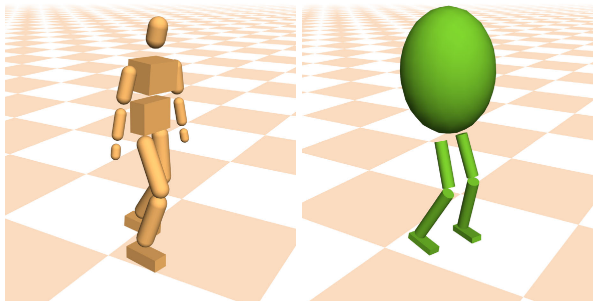

FIGURE 8. The *Humanoid* (left) and *Simple biped* (right) characters.

The *Humanoid* character is similar to the one used in [32]. It has 42 DOFs with 15 links of 95 Kg. $\mathbf{s}_d$ and $\mathbf{a}_p$ are designed similar to those of the *Atlas* character and the target angles of the remaining upper-body joints are defined in the same way as the *Atlas*. This character uses the same FSM-based action range as the *Atlas* character.

The *Simple biped* character has 18 DOFs with 7 links of 70 Kg. $\mathbf{s}_d$ includes the states of the root and lower-body joints and the position of two end-effectors (the left and right feet). $\mathbf{a}_p$ consists of the target angles for the lower-body joints. This character uses the same FSM-based action range as the *Atlas* character except for the action range of the back joint.

VI. LIMITATIONS

As shown in the accompanying video, the learned policy does not create a gait that matches the given desired gait parameters very accurately. We guess one of the reasons for this is that the policy network outputs the target pose to achieve the desired gait parameters, but it is further modified by the balance feedback. It is possible that action selection based on the character state at the beginning of a step may not sufficiently consider the future modification of the target pose. It would be helpful to apply strong levels of balance feedback when there are external perturbations, and weak feedback when maintaining a steady walking state to prioritize achieving the desired parameters. As another reason, the proposed reward scaling method can "balance" the rewards of steps with different duration, but cannot prevent the policy from taking shorter steps to create a greater number of steps during a given simulation time. To handle this problem, we need to consider further how to reflect the simulated step duration in the reward so that the policy does not learn only how to increase the simulated duration of the current step.

We manually set the range of the desired gait parameters to avoid spending time learning impossible combinations of parameters and to reflect the correlation between the parameters in an actual human step. Methods to automatically determine which part of the parameter space to learn intensively, such as adaptive sampling [7], [24] or curriculum learning in the parameter space [12], could be a more generalized approach to tackle this issue. Another possible approach is to sample feasible sets of gait parameters according to the distribution of the gait analysis data of human subjects publicly available from a previous study [33]. By adopting these methods, the agent can effectively explore the gait parameter space during the learning process, which will give us more control over the simulated character with more gait parameters, such as the facing or moving direction.

It is difficult to directly compare the learning speed of other DRL-based methods with ours because there are many different factors such as algorithms, hyperparameters, environments, PL for implementation, simulation time length of an experience tuple and so on. However, the number of experience tuples is comparable with other motion-free methods (Ours: about 1.7×10^7, Peng *et al.* [26]: 1×10^7, Xie *et al.* [12]: 6×10^7, Yu *et al.* [10]: 1×10^7 to 3×10^7), which means that the data efficiency is similar to that of previous studies. However, it is also true that the wall time is longer than that in the other studies (Ours: about 72 hours, Peng *et al.* [26]: 20 hours, Xie *et al.* [12]: 12 to 24 hours, Yu *et al.* [10]: 4 to 15 hours). This may be partly due to the fact that all of the code is written in Python and is insufficiently optimized, and thus unnecessary calculations may waste time in collecting experience tuples. Insufficiently tuned hyperparameters within a limited time could also have an effect.

VII. CONCLUSION AND FUTURE WORK

We presented a DRL framework for learning a FSM-based policy that produces diverse gait styles, as directed by user-specified gait parameters. Whereas most of the previous DRL-based controllers are trained to find actions for all phases across a locomotion cycle, our approach allows the agent to learn actions for only two moments in a cycle, the beginning of each step. This characteristic, along with the feedback law embedded in the FSM-based control algorithm, allows our policy to learn robust locomotion. The learned policy allows the simulated character to walk as instructed even by the continuously changing gait parameters while responding to external perturbations.

We think that applying additional objectives to the reward, such as minimizing head movements or angular momentum [15], could improve the quality of the motions generated by our learned controller. Replacing a rigid-body articulated model with a full-body musculoskeletal model [34], [35] and learning the activation of all muscle-tendon units would result in a control policy that is more energy-efficient in terms of metabolic energy expenditure. Efficient motion-free learning of an energy-efficient locomotion policy that can control a wider range of gait parameters will be an interesting future research topic.

REFERENCES

[1] Y. Lee, S. Kim, and J. Lee, "Data-driven biped control," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 1–8, Jul. 2010.

[2] L. Liu, K. Yin, M. V. D. Panne, and B. Guo, "Terrain runner: Control, parameterization, composition, and planning for highly dynamic motions," *ACM Trans. Graph.*, vol. 31, no. 6, p. 154, 2012.

[3] K. Yin, K. Loken, and M. Van de Panne, "Simbicon: Simple biped locomotion control," *ACM Trans. Graph.*, vol. 26, no. 3, p. 105, Jul. 2007.

[4] S. Coros, P. Beaudoin, and M. V. D. Panne, "Generalized biped walking control," *ACM Trans. Graph.*, vol. 29, no. 4, p. 130, 2010.

[5] T. Kwon, Y. Lee, and M. Van De Panne, "Fast and flexible multilegged locomotion using learned centroidal dynamics," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 1–46, Jul. 2020.

[6] X. Bin Peng, P. Abbeel, S. Levine, and M. van de Panne, "DeepMimic: Example-guided deep reinforcement learning of physics-based character skills," 2018, *arXiv:1804.02717*. [Online]. Available: http://arxiv.org/abs/1804.02717

[7] S. Park, H. Ryu, S. Lee, S. Lee, and J. Lee, "Learning predict-and-simulate policies from unorganized human motion data," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–11, Nov. 2019.

[8] K. Bergamin, S. Clavet, D. Holden, and J. R. Forbes, "DReCon: Data-driven responsive control of physics-based characters," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 206:1–206:11, Nov. 2019.

[9] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. Ali Eslami, M. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," Jul. 2017, *arXiv:1707.02286*. [Online]. Available: http://arxiv.org/abs/1707.02286

[10] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy Locomotion," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 144:1–144:12, Jul. 2018.

[11] Y. Jiang, T. V. Wouwe, F. D. Groote, and C. K. Liu, "Synthesis of biologically realistic human motion using joint torque actuation," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 72:1–72:12, Jul. 2019.

[12] Z. Xie, H. Yu Ling, N. Hee Kim, and M. van de Panne, "ALLSTEPS: Curriculum-driven learning of stepping stone skills," May 2020, *arXiv:2005.04323*. [Online]. Available: http://arxiv.org/abs/2005.04323

[13] M. H. Raibert and J. K. Hodgins, "Animation of dynamic legged locomotion," *ACM SIGGRAPH Comput. Graph.*, vol. 25, no. 4, pp. 349–358, Jul. 1991.

[14] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien, "Animating human athletics," in *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1995, pp. 71–78.

[15] M. Jack Wang, J. D. Fleet, and A. Hertzmann, "Optimizing walking controllers," in *Proc. ACM SIGGRAPH Asia Papers (SIGGRAPH Asia)*. New York, NY, USA: ACM, 2009, pp. 168:1–168:8.

[16] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Optimizing walking controllers for uncertain inputs and environments," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 1–8, Jul. 2010.

[17] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Auto. Robots*, vol. 35, nos. 2–3, pp. 161–176, Oct. 2013.

[18] I. Mordatch, M. D. Lasa, and A. Hertzmann, "Robust physics-based locomotion using low-dimensional planning," *ACM Trans. Graph.*, vol. 29, no. 4, p. 71, 2010.

[19] T. Kwon and J. K. Hodgins, "Momentum-mapped inverted pendulum models for controlling dynamic human motions," *ACM Trans. Graph.*, vol. 36, no. 4, p. 1, Jul. 2017.

[20] L. Liu, M. V. D. Panne, and K. Yin, "Guided learning of control graphs for physics-based characters," *ACM Trans. Graph.*, vol. 35, no. 3, pp. 1–14, Jun. 2016.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, G. M. Bellemare, A. Graves, M. Riedmiller, K. A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[22] L. Liu and J. Hodgins, "Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 142:1–142:14, Jul. 2018.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," Aug. 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347

[24] J. Won and J. Lee, "Learning body shape variation in physics-based characters," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 207:1–207:12, Nov. 2019.

[25] S. Coros, P. Beaudoin, and M. van de Panne, "Robust task-based control policies for physics-based characters," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–9, Dec. 2009.

[26] X. B. Peng, G. Berseth, and M. Van de Panne, "Terrain-adaptive locomotion skills using deep reinforcement learning," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 81:1–81:12, 2016.

[27] S. Ha. (2016). *Pydart2*. [Online]. Available: https://github.com/sehoonha/pydart2

[28] J. Tan, K. Liu, and G. Turk, "Stable Proportional-Derivative Controllers," *IEEE Comput. Graph. Appl.*, vol. 31, no. 4, pp. 34–44, Feb. 2011.

[29] I. Kostrikov. (2018). *Pytorch Implementations of Reinforcement Learning Algorithms*. [Online]. Available: https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[31] DART. (2016). *Fullbody1.skel*. [Online]. Available: https://github.com/dartsim/dart/blob/master/data/skel/fullbody1.skel

[32] Y. Lee and T. Kwon, "Performance-based biped control using a consumer depth camera," *Comput. Graph. Forum*, vol. 36, no. 2, pp. 387–395, 2017.

[33] Y. Lee, K. Lee, S.-S. Kwon, J. Jeong, C. O'Sullivan, M. S. Park, and J. Lee, "Push-recovery stability of biped locomotion," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 180:1–180:9, 2015.

[34] Y. Lee, M. S. Park, T. Kwon, and J. Lee, "Locomotion control for many-muscle humanoids," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 218:1–218:11, 2014.

[35] S. Lee, M. Park, K. Lee, and J. Lee, "Scalable muscle-actuated human simulation and control," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 73:1–73:13, Jul. 2019.

GYOO-CHUL KANG received the B.S. degree in computer software from Kwangwoon University and the M.S. degree in computer science from Hanyang University. He is currently a Software Engineer with CLO Virtual Fashion LLC. His research interests include character motion synthesis, physically simulated character, and deep reinforcement learning.

YOONSANG LEE received the Ph.D. degree in computer science and engineering from Seoul National University. He is currently a Professor of computer science with Hanyang University. His research interest includes controlling movements of natural or artificial creatures in from virtual environment to real world.