Self-Imitation Learning of Locomotion Movements through Termination Curriculum

Amin Babadi
amin.babadi@aalto.fi
Department of Computer Science
Aalto University
Helsinki, Finland

Kourosh Naderi
kourosh.naderi@aalto.fi
Department of Computer Science
Aalto University
Helsinki, Finland

Perttu Hämäläinen
perttu.hamalainen@aalto.fi
Department of Computer Science
Aalto University
Helsinki, Finland

ABSTRACT

Animation and machine learning research have shown great advancements in the past decade, leading to robust and powerful methods for learning complex physically-based animations. However, learning can take hours or days, especially if no reference movement data is available. In this paper, we propose and evaluate a novel combination of techniques for accelerating the learning of stable locomotion movements through self-imitation learning of synthetic animations. First, we produce synthetic and cyclic reference movement using a recent online tree search approach that can discover stable walking gaits in a few minutes. This allows us to use reinforcement learning with Reference State Initialization (RSI) to find a neural network controller for imitating the synthesized reference motion. We further accelerate the learning using a novel curriculum learning approach called Termination Curriculum (TC), that adapts the episode termination threshold over time. The combination of the RSI and TC ensures that simulation budget is not wasted in regions of the state space not visited by the final policy. As a result, our agents can learn locomotion skills in just a few hours on a modest 4-core computer. We demonstrate this by producing locomotion movements for a variety of characters.

CCS CONCEPTS

• **Computing methodologies** → **Reinforcement learning**; **Procedural animation**; *Robotic planning*; *Physical simulation*; Continuous space search; Neural networks.

KEYWORDS

Continuous Control, Physically-Based Animation, Online Optimization, Reinforcement Learning, Policy Optimization, Self-Supervised Learning

ACM Reference Format:

Amin Babadi, Kourosh Naderi, and Perttu Hämäläinen. 2019. Self-Imitation Learning of Locomotion Movements through Termination Curriculum. In *Motion, Interaction and Games (MIG '19)*, October 28–30, 2019, Newcastle upon Tyne, United Kingdom. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3359566.3360072>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MIG '19, October 28–30, 2019, Newcastle upon Tyne, United Kingdom

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6994-7/19/10...\$15.00

<https://doi.org/10.1145/3359566.3360072>

1 INTRODUCTION

Intelligent control of physics simulation is an increasingly popular approach for synthesizing physically-plausible animations for simulated characters. This requires a method that outputs, for each timestep, simulation actuation parameters such as joint torques such that the character performs some desired movement. This poses a continuous control problem with high state and control dimensionality, where the environment is governed by complex physical interactions.

Current approaches for solving physically-based animation can be divided into two categories: 1) planning and search, and 2) reinforcement learning. Planning and search methods use the interleaved mechanism of iterative generation and evaluation of candidate solutions until finding a sufficiently good one. On the other hand, reinforcement learning (RL) methods learn how to act through interaction with the environment.

In this paper, we propose a self-imitation learning approach for enabling rapid learning of stable locomotion controllers. Essentially, our approach combines FDI-MCTS [Rajamäki and Hämäläinen 2018] and DeepMimic [Peng et al. 2018a], two recent methods for continuous control. We are motivated to mitigate the main limitations of both methods, namely the high run-time cost of FDI-MCTS and the data-dependency and sample complexity of DeepMimic.

We begin by using FDI-MCTS to generate a cyclic locomotion movement as the reference motion. Then we employ a training mechanism similar to DeepMimic, to find a neural network controller that imitates the reference motion. We also propose Termination Curriculum (TC), a novel curriculum learning approach for accelerating the imitation learning. Our experiments show that our approach is able to learn robust locomotion skills for a broad set of 3D characters. All controllers are trained in less than four hours, which is significantly faster than DeepMimic and SFV.

2 RELATED WORK

Our work aims at producing locomotion movements for physically-based characters. There has been a large body of research on this problem, especially after remarkable breakthroughs in Deep Reinforcement Learning (DRL) [Mnih et al. 2015; Silver et al. 2016, 2017]. Proposed approaches can be divided into two categories: planning and search (Section 2.1), and reinforcement learning (Section 2.2).

2.1 Planning and Search

Evolutionary Strategies (ES) are a family of black-box optimization methods that are also very easy to use in parallel [Salimans et al. 2017]. One of the most common ES methods is Covariance Matrix

Adaptation Evolution Strategy (CMA-ES) [Hansen 2006]. CMA-ES has been used in an offline manner to learn the parameters for controller of physically-simulated characters [Geijtenbeek et al. 2013]. Another study has used CMA-ES as an offline low-level controller for synthesizing humanoid wall climbing movements [Naderi et al. 2017]. Recent studies have used CMA-ES for synthesizing sports movements in a two-player martial arts interface [Babadi et al. 2018] and single-agent basketball [Liu and Hodgins 2018].

Monte Carlo methods have received considerable interest in domains where the search budget is limited. Sequential Monte Carlo (SMC) has been shown to be effective for online synthesis of physically-based animations [Hämäläinen et al. 2014]. It has also been used for replicating motion capture data by breaking the problem into a sequence of control fragments [Liu et al. 2016]. Another Monte Carlo method, called Monte Carlo Tree Search (MCTS) [Browne et al. 2012], has shown great performance in real-time applications and games [Sironi et al. 2018]. MCTS has been used in Alpha Go Zero for improving the policy using self-play [Silver et al. 2017]. Fixed Depth Informed MCTS (FDI-MCTS) is a continuous version of MCTS used in physically-based control. FDI-MCTS uses a policy network trained with supervised learning to reduce the movement noise caused by the sampling-based controller [Rajamäki and Hämäläinen 2018].

2.2 Reinforcement Learning

Reinforcement Learning (RL) methods have become significantly more powerful in the recent years, mainly after the Deep Reinforcement Learning (DRL) success in Atari games [Mnih et al. 2015]. Two of the most common RL algorithms are called Trust Region Policy Optimization (TRPO) [Schulman et al. 2015] and Proximal Policy Optimization (PPO) [Schulman et al. 2017]. The key element in these methods is a surrogate objective function that allows for more than one gradient update per data sample. Studies have shown that PPO outperforms TRPO in most cases, which makes it the dominant algorithm used in continuous control [Peng et al. 2018a; Yu et al. 2018].

Curriculum learning is the process of learning a series of tasks in increasing order of complexity [Bengio et al. 2009]. It is a powerful technique for improving learning performance in terms of both convergence speed and output quality. This technique has been used for learning humanoid climbing movements, where the agent learns 1-, 2-, 3-, and 4-limb movements in order (a limb can be either one of agent's hands or feet) [Naderi et al. 2018]. A recent study proposed a continuous curriculum learning method for providing physical assistance to help the character in locomotion movements [Yu et al. 2018].

A popular data-driven approach for solving physically-based control is to learn through imitation. An example in this category is called DeepLoco, a system that trains high-level and low-level controllers such that the low-level controller is encouraged to imitate a reference animation [Peng et al. 2017]. A descendant of this method, called DeepMimic, has been able to produce a wide set of high-quality and robust movements by imitating a large motion capture dataset [Peng et al. 2018a]. The most recent variant of this method is able to extract reference animations directly from videos,

which makes the training pipeline significantly cheaper [Peng et al. 2018b].

3 PRELIMINARIES

Online optimization is one of the main approaches for generating physically-based animations [Geijtenbeek and Pronost 2012]. The idea is to generate a set of candidate solutions (i.e., action sequences), and find the most cost-efficient solution by evaluating them. This is usually done using forward simulation until some time horizon and computing some cost function that encodes some information about the target movement. In the walking task for example, the cost function encourages the character to keep its center of mass above its feet and its mean velocity close to some desired walking velocity. The function also usually penalizes the amount of torque applied to each body joint in order to avoid unrealistic movements [Rajamäki and Hämäläinen 2017].

We use a recent open-source¹ online optimization method called *Fixed-Depth Informed Monte Carlo Tree Search (FDI-MCTS)* [Rajamäki and Hämäläinen 2018] to produce a cyclic locomotion movement as the reference motion. FDI-MCTS synthesizes movements using an interleaved process of tree search and supervised learning. The supervised learning component is trained using the best controls found during the tree search in order to reduce the noise in produced movements. This results in emergence of stable locomotion gaits for different types of characters in less than a minute, allowing rapid cost function design iteration. Movement is initially noisy, but running the algorithm for a few more minutes removes the noise.

The basic definition of reinforcement learning includes an agent that has interactions with some environment, and its goal is to maximize the accumulated rewards over time [Sutton and Barto 2018]. Policy optimization refers to a family of reinforcement learning methods, in which the goal is to optimize the agent's policy with respect to the expected return. The policy is usually modelled using a neural network, and defines a mapping from a state to a distribution over actions.

In this paper, we use an open-source² implementation of *Proximal Policy Optimization (PPO)* [Schulman et al. 2017]. PPO uses stochastic gradient ascent by estimating the gradient of the expected return with respect to the policy parameters. It does that using a so-called *clipped surrogate objective function*, that penalizes large policy updates.

4 METHOD

4.1 Reference Motion Generation

The first step of our approach includes automatic generation of a reference motion. For this purpose, we use FDI-MCTS [Rajamäki and Hämäläinen 2018], a recent sampling-based model-predictive algorithm for continuous control. The cost function used by FDI-MCTS uses four quadratic terms for penalizing the followings: 1) amount of torques applied to joints, 2) deviation from the default pose (shown in Fig. 1), 3) planar deviation of the center of mass from the feet mean point, and 4) the difference between the current and the target velocity of the character. When the locomotion gait

¹<https://github.com/JooseRajamaeki/TVCG18>

²<https://github.com/openai/baselines>

Figure 1: Simulated 3D characters and their physical skeletons in the default pose. Since our approach does not require any hand-designed or mocap animations, it can be applied to a wide range of character anatomies with minimum cost.

has become stable (after a few minutes in our experiments), we extract a cycle out of the synthesized motion sequence using the method explained next.

Given a trajectory of locomotion movements, the cycle extraction process starts by storing the key information at every timestep. Stored information include orientation q_t^j of each joint j , angular velocity $\dot{q}_t^b$ of each bone b , position p_t^e and linear velocity v_t^e of each end-effector e (e.g., hands and feet for humanoid characters), and the character’s center of mass p_t^{com} . At each timestep, in order to detect the end of the cycle, positions and linear velocities of all end-effectors are compared with their corresponding values at the initial timestep of the cycle. The cycle is completed if the end-effectors have almost the same positions as in the initial timestep and their corresponding linear velocities have acute angles between them, i.e., their dot product is positive. We set the minimum length of a cycle to 10 timesteps to avoid detecting empty cycles. This gives us an easy-to-implement and computationally cheap method for extracting cycles from synthesized movements.

4.2 Self-Imitation Learning

After synthesizing a cyclic movement, we use PPO to find a policy for performing locomotion while imitating the reference motion. In this part, we use a training mechanism similar to DeepMimic [Peng et al. 2018a]. When starting a new episode, the so-called *Reference State Initialization (RSI)* is used, i.e., the initial state is uniformly picked from the reference motion. An episode is terminated if a bone other than the feet is in contact with the ground, or episode length exceeds some pre-defined limit. To accelerate the training process, we employ a different *Early Termination (ET)* mechanism, which is explained in Section 4.4.

4.3 Reward Function

Our reward definition is almost identical to DeepMimic [Peng et al. 2018a]. The instantaneous reward r_t at timestep t is defined as follows:

$$r_t = \omega^I r_t^I + \omega^T r_t^T,$$

where r^I (weighted by ω^I) and r^T (weighted by ω^T) define the imitation and task rewards at timestep t . This encourages the character to satisfy the task objective while imitating the reference motion.

All quantities on the right side of the equation are between 0 and 1, and $\omega^I + \omega^T = 1$, leading to a bounded reward $0 \leq r_t \leq 1$.

4.3.1 Imitation Reward. The imitation reward encourages the character to imitate the reference motion, and it is computed as a weighted sum of four terms as follows:

$$r_t^I = w^p r_t^p + w^v r_t^v + w^e r_t^e + w^c r_t^c$$

$$w^p = 0.65, w^v = 0.1, w^e = 0.15, w^c = 0.1$$

The terms r^p , r^v , r^e , and r^c are computed exactly the same as in [Peng et al. 2018a].

4.3.2 Task Reward. In this paper, we only consider the locomotion tasks because FDI-MCTS is mainly designed to produce locomotion movements. So the task reward encourages the character to walk in the desired direction with the desired speed, and it is defined as follows:

$$r_t^T = \exp \left[-2.5 \times \|v^* - \bar{v}\|^2 \right],$$

where v^* is the desired velocity and $\bar{v}$ is the mean velocity of character’s bones, projected on the xy-plane.

4.4 Termination Curriculum

When using RL algorithms, a common challenge appears in the initial stages of the training, where the policy can easily lead the agent to the fruitless regions of the state space. Plus, it has been recently shown that PPO is prone to getting stuck in local optima [Hämäläinen et al. 2018]. These can cause a huge waste of simulation budget during the training. We use a didactic example, shown in Fig. 2 to demonstrate this problem and how we propose to solve it.

Fig. 2b, shows how the simulation budget can be wasted even when *Reference State Initialization (RSI)* is used without any termination mechanisms. The figure shows a 2D state space, in which the optimal state trajectory (i.e., the reference motion) is shown in dark gray. The light-to-dark heatmap shows the state-dependent reward distribution and each solid black line resembles a random trajectory (i.e., episode) in the state space. As it can be seen in the figure, RSI forces the episodes to start on the optimal state trajectory. However, a non-optimal policy can easily lead the agent to the regions of the state space that should not be visited using an optimal policy.

Figure 2: This didactic example demonstrates the movement of an agent along the vertical axis over time (the horizontal axis). The reward landscape is shown using a light-to-dark heatmap, i.e., the optimal state trajectory is the dark sine curve. The solid black lines are the observed trajectories, whose initial states are shown by white circles. Fixed state initialization can slow down the training process since a slight deviation from the optimal trajectory can easily prevent the agent from visiting promising regions of the state space (a). *Reference State Initialization (RSI)* [Peng et al. 2018a] is an effective strategy for mitigating this issue (b). We propose termination curriculum (TC), an early termination strategy, by putting a threshold (shown by the red dashed lines) on the amount of instantaneous reward at each timestep (c). Using this strategy, at early stages of the training the agent will end up observing a lot of short trajectories since the policy is very likely to deviate from the optimal trajectory. However, as the training goes on, the agent will learn to stay in proximity of the optimal trajectory for a longer time, making it possible to reduce the reward threshold (d).

To mitigate the problem shown in Fig. 2b, we propose *Termination Curriculum (TC)*. TC is a continuous curriculum learning mechanism that limits the minimum allowed amount of instantaneous reward in each timestep, and then lowers this limit during training. This simple mechanism can lead to a significant increase of performance in terms of both reward and training speed. Next we explain how TC works in detail.

Let R_t^{min} denote the threshold for the instantaneous reward at timestep t . We add an extra termination condition to the underlying MDP such that an episode is terminated if $r_t < R_t^{min}$. This forces the agent to only visit the regions of the state space in which $r_t \geq R_t^{min}$, except in the last timestep of each episode. By lowering the threshold R_t^{min} during the training process, the agent is gradually allowed to visit other regions of the state space as well. In other words, the character performs in a restricted state space, which in the beginning is significantly smaller than the original space. By gradually relaxing the restrictions, the state space becomes larger, allowing the character to learn how to act in the new states.

Fig. 2c demonstrates how termination curriculum mechanism helps the agent to cover the states that are in proximity to the optimal state trajectory. In this case, the trajectories are shorter. However, when used along with RSI, most of the simulation budget is spent for revising the policy in proximity of the optimal state trajectory. Furthermore, lowering the reward threshold allows the agent to visit more challenging regions of the state space, extending the policy to act in longer episodes.

The main challenge when applying termination curriculum is how to choose the right range for R_t^{min} . As explained in Section 4.3, our DeepMimic-style reward function is human-understandable enough, and it is always between 0 and 1. Our tests show that starting with $R_t^{min} = 0.75$ and linearly decreasing it to $R_t^{min} = 0.5$ produces good results for a wide range of characters.

5 EVALUATION

5.1 Implementation Details

We now explain the implementation details for producing the results. The source code is available on GitHub³, and examples of synthesized motions can be seen in the supplemental video⁴.

Physical Simulations: We used Open Dynamics Engine (ODE) [Smith 2001] for physical simulations. Simulations were done in 64 parallel threads in order to accelerate the optimization and training processes.

State Features: At each timestep t , the state features contain position and orientation of the root bone, angular velocity of each bone, and finally, all the joint angles. These values are concatenated into a vector and used as the feature vector s_t .

Action Parameterization: Action parameters are defined as the reference angles α_{ref} for each degree of freedom. A P-controller then converts these values to reference angular velocities, i.e., $\omega_{ref} = K_p (\alpha_{ref} - \alpha_{cur})$, where $K_p = 10$ is the P-controller’s multiplier and α_{cur} is the current angle.

Training: We used Tensorflow [Abadi et al. 2015] for training the agent using the PPO algorithm [Schulman et al. 2017]. Policy and value functions were modeled using fully-connected networks with two hidden layers of size 64 with tanh activation function. Our tests showed that it is better to use tanh in the final layer of the policy network and then interpolate the policy’s output using the minimum and maximum angle for each degree of freedom.

Training Parameters: The parameters used for PPO training are shown in Table 1.

³<https://github.com/donamin/self-imitation-learning>

⁴<https://youtu.be/dxTZk35Ofyg>

Table 1: Parameters used in the PPO training

Parameter	Value
Clipping coefficient (ϵ)	0.2
Number of epochs per iteration	25
Learning rate*	$10^{-4}, 10^{-7}$
Training iterations	3500
Iteration simulation budget	4096
Batch size	256
Discount factor (γ)	0.99
GAE parameter (λ)	0.95

* Learning rate decay was used throughout the training.

5.2 Experiments

5.2.1 Characters. In order to show the flexibility of our approach, we used a variety of game characters in our simulations⁵. The character's physical skeletons were modeled using 3D capsules connected via 3-DOF ball-and-socket or 1-DOF hinge joints. Finally, characters were rendered using Unity 3D game engine⁶. Fig. 1 shows all characters used in our work along with their skeletons. The details of these characters in the physical simulations are shown in Table 2.

Table 2: Setup details of the simulated characters

Property	Wolf	Orc	Mech
Height (m)	1.3	1.6	2.1
Mass (kg)	50	30	30
Bones	17	13	8
Joints	16	12	7
State Dimensions	87	69	37
Action Dimensions (DoF)	30	24	7

5.2.2 Experiments. We used all characters shown in Fig. 1 for solving the locomotion task, i.e., moving in the forward direction with the target speed of 1m/s.

Reference motions: The walking cycles produced in the reference motion generation stage had usually between 20 to 40 frames (depending on the character). Thus, the maximum episode length during the training was set to 100 to ensure that each episode gives the characters enough time for repeating the reference motion at least twice.

Termination curriculum: In order to show the effectiveness of the termination curriculum mechanism, we tested five different termination strategies as follows (all versions use reference state initialization):

- (1) **No termination:** This version does not use any termination strategies (similar to Fig. 2b).
- (2) **Termination curriculum:** In this version, the training begins by setting the threshold R_t^{min} (introduced in Section 4.4) to 0.75 and then linearly decaying it to 0.5 throughout

the training. The next three versions are solely introduced to demonstrate the effectiveness of decaying the threshold and thus use constant values for the threshold R_t^{min} .

- (3) **Tight threshold:** This version uses the constant threshold $R_t^{min} = 0.75$ for episode termination. This tight threshold only allows the agent to visit the states that are "almost perfect" (similar to Fig. 2c).
- (4) **Medium threshold:** In this version, the constant threshold $R_t^{min} = 0.5$ is used to limit the visible states to those with a fairly good reward value (similar to Fig. 2d).
- (5) **Loose threshold:** Finally, a version with the constant threshold $R_t^{min} = 0.25$ was defined. This threshold does not allow the agent to visit states with very bad rewards, but does not guarantee to keep it in good states.

Reward validation: In order to show the effectiveness of the reward function defined in Section 4.3, we also trained the agents directly using the FDI-MCTS cost function $\mathcal{J}_t$ converted to reward as:

$$r_t = \frac{b - \mathcal{J}_t}{b},$$

where $b = 10000$ is a constant used to map the cost values into the range $[0, 1]$. Note that the reward function gradient is proportional to FDI-MCTS objective function, thereby preserving the online optimization landscape.

5.3 Results

Each one of the five versions introduced in Section 5.2.2 was tested in five independent runs, and the mean and standard deviation of average cost and reward were recorded. In each run, a new walking cycle was generated using FDI-MCTS and then trained using PPO algorithm for 5000 iterations. All experiments were performed by an Intel Core i7-4930K 3.40GHz CPU processor and 16GB of RAM. Examples of synthesized locomotion movements can be seen in the supplemental video⁷.

5.3.1 FDI-MCTS is efficient in producing reference motions. All reference motions were produced in less than five minutes. The initial and final frames of the cycles produced in the reference motion generation stage are pretty similar, although not exactly the same. However, slight differences are not a problem since in the self-imitation learning stage, the agent tries to imitate the reference cycle as much as the reward function allows it to. If the reference cycle is not perfect (which happens in most of the cases), the agent compensates the gap between initial and final frames such that the maximum reward in both frames is produced. This results in a motion that resembles the reference cycle as much as possible.

5.3.2 Termination Curriculum improves training. The plots in Fig. 3 show how different termination strategies work in terms of average reward. Compared to other versions, the termination curriculum shows superior performance for all characters except the wolf, where its performance is similar to the tight threshold strategy. The reason is that the wolf character, unlike orc and mech, is a quadruped and in the case of quadrupeds, a wide range of policies (including the random policies) can easily avoid the character from

⁵All 3D models are royalty-free assets purchased from Unity Asset Store.

⁶<https://www.unity3d.com/>

⁷<https://youtu.be/dxTZk35Ofyg>

Figure 3: Evaluating different termination strategies. Termination Curriculum (TC) significantly accelerates the training, allowing to train characters with various anatomies in only a few hours.

Figure 4: Comparing the performance, in terms of FDI-MCTS cost function, when using the DeepMimic-style reward function explained in Section 4.3 instead of directly using FDI-MCTS cost function as reward.

falling down. This results in receiving high rewards in Fig. 3a, and the curriculum has less effect. However, as it can be seen for the orc (Fig. 3b) and mech (Fig. 3c) characters, the tight threshold strategy shows instabilities since it uses a very optimistic reward threshold, which makes it fragile to small noise in the environment. On the other hand, the termination curriculum is producing stable results over all five runs for the three characters.

Another observation in Fig. 3 is how the termination curriculum significantly decreases the sample complexity when compared to the naive version with no termination. This, together with multi-threading, enabled us to successfully train the agents in only a few hours (including the reference motion generation stage). This is very promising since it can be used by game developers and animators as an easy-to-use and cheap animation production pipeline.

5.3.3 The combination of FDI-MCTS and PPO is better than PPO alone. Fig. 4 plots the FDI-MCTS cost when using PPO with DeepMimic-style reward function r_t , explained in Section 4.3, as opposed to the reward function $r_t = \frac{b - \mathcal{J}_t}{b}$, which directly optimizes the FDI-MCTS objective function $\mathcal{J}_t$, while at the same time avoiding very large and small rewards, which is required for PPO’s value function predictor network training. As it can be seen in the figure, the DeepMimic-style reward acts as a good proxy for optimizing the FDI-MCTS cost. On the other hand, trying to directly apply PPO to the locomotion problem without the FDI-MCTS imitation reward leads to clearly inferior results.

6 DISCUSSION AND CONCLUSIONS

We proposed an approach for constructing a policy network for synthesizing stable locomotion movements for arbitrary character anatomies. Our approach starts by running an online optimization method (FDI-MCTS [Rajamäki and Hämäläinen 2018]) to generate a stable locomotion gait. It then extracts a cyclic motion out of the generated movement as the reference motion. In the next stage, inspired by DeepMimic [Peng et al. 2018a], proximal policy optimization (PPO) [Schulman et al. 2017] is used such that the character is able to accomplish the locomotion task while imitating the reference motion. We also proposed Termination Curriculum (TC), a continuous curriculum learning mechanism to enable rapid training of the final policy. The core idea of this mechanism is to terminate the episode if the instantaneous reward becomes lower than some threshold R_t^{min} . Decreasing this threshold during the training results in a continuous curriculum which limits the state space regions that are visible to the agent during training.

Our experiments show that the proposed FDI-MCTS to PPO pipeline combines the advantages of both algorithms. FDI-MCTS allows rapid discovery and visualization of behaviors, enabling fast reward function design iteration. FDI-MCTS also provides more flexibility in the reward design, as one can simply use quadratic cost terms without worrying about excessive reward magnitude. Once a suitable gait has been found, PPO with DeepMimic reward function can produce a stable and computationally efficient neural network policy. In contrast, using FDI-MCTS alone incurs orders

of magnitude higher runtime cost due to the forward simulation, and PPO alone prevents the fast reward design iteration. In absence of the FDI-MCTS -generated reference motion, PPO also failed in optimizing our locomotion reward function.

Although our approach improves the sample complexity of DeepMimic-style learning, it still has a high sample complexity. In future work, this could be improved by using more recent state-of-the-art reinforcement learning algorithms, such as Maximum a Posteriori Policy Optimisation (MPO) [Abdolmaleki et al. 2018] or Soft Actor-Critic (SAC) [Haarnoja et al. 2018]. However, even with a more advanced RL algorithm, our proposed approach of using a trajectory optimization method for reference movement generation will probably offer faster reward function and movement style design iteration, as opposed to using RL alone.

ACKNOWLEDGMENTS

This work has been supported by the Academy of Finland grants 299358 and 305737. The authors would like to thank the reviewers for their useful comments for improving this manuscript. Amin Babadi additionally thanks JooSe Rajamäki for his useful discussions and continuous support.

REFERENCES

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. <https://www.tensorflow.org/> Software available from tensorflow.org.
- Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. 2018. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920* (2018).
- Amin Babadi, Kourosh Naderi, and Perttu Hämäläinen. 2018. Intelligent middle-level game control. In *Proceedings of IEEE Conference on Computational Intelligence and Games (IEEE CIG)*. IEEE.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of International Conference on Machine Learning (ICML)*. ACM, 41–48.
- Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.
- Thomas Geijtenbeek and Nicolas Pronost. 2012. Interactive character animation using simulated physics: A state-of-the-art review. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2492–2515.
- Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. 2013. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)* 32, 6 (2013).
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290* (2018).
- Perttu Hämäläinen, Amin Babadi, Xiaoxiao Ma, and Jaakko Lehtinen. 2018. PPO-CMA: Proximal Policy Optimization with Covariance Matrix Adaptation. *arXiv preprint arXiv:1810.02541* (2018).
- Perttu Hämäläinen, Sebastian Eriksson, Esa Tanskanen, Ville Kyrki, and Jaakko Lehtinen. 2014. Online motion synthesis using sequential monte carlo. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 51.
- Nikolaus Hansen. 2006. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation*. Springer, 75–102.
- Libin Liu and Jessica Hodgins. 2018. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 142.
- Libin Liu, Michiel Van De Panne, and KangKang Yin. 2016. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics (TOG)* 35, 3 (2016), 29.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.
- Kourosh Naderi, Amin Babadi, and Perttu Hämäläinen. 2018. Learning Physically Based Humanoid Climbing Movements. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 69–80.
- Kourosh Naderi, JooSe Rajamäki, and Perttu Hämäläinen. 2017. Discovering and synthesizing humanoid climbing movements. *ACM Transactions on Graphics (TOG)* 36, 4, Article 43 (July 2017), 11 pages. <https://doi.org/10.1145/3072959.3073707>
- Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018a. Deep-Mimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)* 37, 4, Article 143 (July 2018), 14 pages. <https://doi.org/10.1145/3197517.3201311>
- Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 41.
- Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. 2018b. SFV: Reinforcement learning of physical skills from videos. *ACM Transactions on Graphics (TOG)* 37, 6, Article 178 (Nov. 2018), 14 pages.
- JooSe Rajamäki and Perttu Hämäläinen. 2017. Augmenting sampling based controllers with machine learning. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 11.
- JooSe Julius Rajamäki and Perttu Hämäläinen. 2018. Continuous control monte carlo tree search informed by multiple experts. *IEEE transactions on visualization and computer graphics* (2018).
- Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* (2017).
- John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, 1889–1897.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
- David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.
- David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354.
- Chiara F Sironi, Jialin Liu, Diego Perez-Liebana, Raluca D Gaina, Ivan Bravi, Simon M Lucas, and Mark HM Winands. 2018. Self-adaptive mcts for general video game playing. In *International Conference on the Applications of Evolutionary Computation*. Springer, 358–375.
- Russell L. Smith. 2001. Open Dynamics Engine. <http://www.ode.org/>. Accessed: 2019-01-01.
- Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
- Wenhao Yu, Greg Turk, and C Karen Liu. 2018. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 144.