A Reinforcement Learning Approach To Synthesizing Climbing Movements

Kourosh Naderi, Amin Babadi, Shaghayegh Roohi, and Perttu Hämäläinen
Department of Computer Science
Aalto University
Helsinki, Finland
firstname.familyname@aalto.fi

***Abstract*—This paper addresses the problem of synthesizing simulated humanoid climbing movements given the target holds, e.g., by the player of a climbing game. We contribute the first deep reinforcement learning solution that can handle interactive physically simulated humanoid climbing with more than one limb switching holds at the same time. A key component of our approach is Self-Supervised Episode State Initialization (SS-ESI), which ensures diverse exploration and speeds up learning, compared to a baseline approach where the climber is reset to an initial pose after failure. Our results also show that training with a multi-step action parameterization can produce both smoother movements and enable learning from slightly fewer explored actions at the cost of increased simulation time per action.**

***Index Terms*—reinforcement learning, climbing movements, state initialization, action parameterization**

I. Introduction

Deep reinforcement learning (Deep RL) has become increasingly popular since the demonstration of its super-human performance in playing Atari games [1]. Following this, RL has been successfully applied in many contexts, e.g., games [2], physically-based animations [3], [4], and robotics [5]. However, formulating a task in the form of RL and designing a meaningful reward function are challenging and problem-dependent tasks. One of the still unexplored and nontrivial avenues is synthesizing plausible humanoid climbing movements.

Using reinforcement learning to synthesize physically-based movements has been an active area of research for years. However, many state-of-the-art approaches rely on high-quality reference motions such as motion capture animation data, which are expensive or impossible to record depending on the character and the environment. Another limitation of current approaches is their focus on non-extreme movements, such as locomotion, where the agent's interaction with the environment is limited.

This paper addresses the problem of synthesizing climbing movements in simulated indoor bouldering environments based on given target hands/feet hold positions (see Fig. 1). In a bouldering environment, the climbers start on the ground in front of the wall, and they are asked to start by getting hands/feet to some predefined hold(s). The climber's goal is to reach some specified goal hold (usually at the top of the wall) by performing various climbing movements, some of which can be challenging and cumbersome.

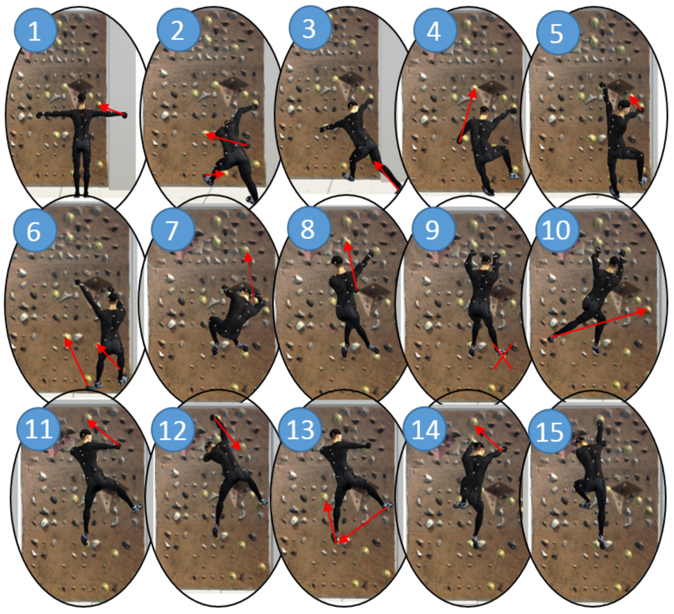

Fig. 1: Climbing movements synthesized by the trained policy on a climbing route. The user specified commands are shown as the red arrows. Red crosses denote letting a limb be not connected with any hold.

This paper proposes a reinforcement learning-based framework for solving humanoid climbing problems in physically simulated environments. Our framework is able to synthesize climbing movements that are physically plausible, and can involve moving more than one-limb at a time or have free limbs (see Fig. 1). The main difficulty of the climbing problem is that the training requires a large amount of simulated experience in order to explore a sufficiently wide range of climbing movements. In order to mitigate this issue, motivated by the success of the reference state initialization approach of [3], we compare two episode state initialization strategies by evaluating their effect on exploring various movements. We also demonstrate that movements with different smoothness can be synthesized using different action parameterizations. Our proposed framework is able to explore a wide range of climbing movements with a high success rate.

The rest of this paper is organized as follows: Section II reviews some of the related work. In Sections III and IV, we introduce our training environment, formulate the climbing problem in the form of reinforcement learning, and investigate an exploration method for better performance in learning climbing movements. Finally, we conclude the paper with discussions on the experiments, results, limitations and possible future work in Sections V, VI and VII, respectively.

II. Related Work

Our main contribution is to formulate and solve the humanoid climbing problem using reinforcement learning. Thus, at first we review the latest RL approaches for learning how to act in environments with continuous state and action spaces. Next, we discuss the state-of-the-art approaches for solving the climbing problem in physically-simulated settings.

A. Reinforcement Learning in Continuous Environments

Reinforcement learning studies the process of learning to act in an environment through trial and error in order to maximize rewards. Thanks to impressive advances in deep neural networks, RL has experienced several major breakthroughs in the last five years [1], [6], [7]. Although much of the work focuses on discrete actions like playing classic Atari games, the state-of-the-art has also advanced tremendously in problems with high-dimensional continuous action spaces [8], which we briefly overview in this part.

Most continuous control RL relies on Monte Carlo gradient estimates computed from simulation data. However, simply following the gradient can be unstable. To mitigate this, trust region family methods have been introduced to solve this issue by limiting the amount of deviation between policies in each update [9]. Proximal Policy Optimization (PPO), a offspring of this family, uses a surrogate loss function instead of a hard limit [10]. PPO has produced remarkable results in a wide range of continuous domains, and has become the default algorithm in the compute graphics community [2], [3]. PPO is also the algorithm used in this work to solve the humanoid climbing problem, using the Unity ML-Agent framework [2]. A more recent approach with state-of-the-art results is called soft actor-critic (SAC), which uses maximum entropy principle to apply off-policy learning [5]. It has been shown that it can be successfully used in real-world robotic problems with reasonable sample complexity [11].

An alternative to policy gradient methods is to formulate policy updates as fitting the policy distribution to the explored actions, each action weighted based on its estimated value or advantage. This bridges RL to iterative distribution fitting optimization methods like CMA-ES. PPO-CMA utilizes distribution adaptation techniques inspired by CMA-ES to solve the premature convergence problem of PPO [12]. A similar approach has been used in Maximum a Posteriori Policy Optimisation (MPO) [13] and its recent variant [14], and has shown great results in DeepMind control suite [15] and OpenAI Gym [16] environments.

B. Synthesizing Climbing Movements

Many previous studies of climbing motion synthesis exist. [17] plans one-limb climbing movement for a four-limbed robot by sampling for center of mass of the robot and obtaining a closed form solution for the moving limb. [18] plans one-limb movement of humanoid climber using a reinforcement learning approach and uses transfer learning to adopt the planning to the new wall, however to move the body inverse kinematics is used. In this paper, we propose a reinforcement learning approach that can be used for an interactive climbing simulation or a game. In contrast to [18] our approach can handle multiple simultaneously moving limbs and the climber can have free limbs which leads to synthesizing more complex movements.

[19], [20] showed that climbing movements can emerge from physically-based movement optimization. However, the character was limited to perform slow and more balanced climbing movements by moving one-limb at a time, e.g., move hands and then legs to climb the wall. [21] introduces a hierarchy of low-level optimization and high-level path planner to synthesize more plausible and dynamic/agile climbing movements limited to moving two limbs at a time. In their work, the agent could decide on alternating between slow and dynamic climbing movements. This work later was enhanced in [22] by utilzing neural network predictions to synthesize more successful transitions and complex movements such as dynamic jumps or dynos. There also exists work that focuses on subproblems such as climber finger and grasping simulation [23]. [24] introduces a method called 2PAC that can synthesize motions including climbing for an arbitrary character by contact planning and a novel formulation of the kinematic constraints, which allows them to generate a quasi-static center of mass trajectory for the character. However, all these optimization approaches are computationally expensive. In contrast, the promise of RL is that although the training phase can be costly, the trained neural network policy can be very lightweight and applicable to resource-constrained real-time applications like computer games.

III. Preliminaries

A. Climbing Environment

Environment: Fig. 2 shows the climbing environment for training. The climber starts from T-pose in front of 16 randomized holds. The position of each hold h is denoted by $\mathbf{x}_h$. The holds are positioned as a 4×4 grid with random perturbation added. The position of free limbs, i.e., limbs that are not attached to a hold, is denoted by $\mathbf{x}_{-1}$. Holds are modeled as spheres with radius of 15cm with ball-and-socket joints that climber's hands/feet can be attached to.

Climbing State: Our climber has $1.75m$ height and $70kg$ weight, although it is trivial to generalize this framework to other character anatomies. The climber has 15 bones with a total of 44 degrees of freedom, which consist of 30 target joint angles and 14 joint strength values for all the bones excluding the root. The state of the climber at time t is denoted by

$\mathbf{s}_t = \{\langle \mathbf{x}_b, \mathbf{v}_b, \mathbf{q}_b, \omega_b, \tau_b, \mathcal{I}_{\text{wall}}(b), \mathcal{I}_{\text{ground}}(b) \rangle \, \forall b \in \mathcal{B}\}$, where $\mathcal{B}$ is the set of all bones, and bone state comprises 3D position, linear velocity, rotation quaternion, angular velocity, and strength. The strength τ_b corresponds to the maximum torque parameter of the physics engine's joint motor that controls bone b. Two indicator functions $\mathcal{I}_{\text{wall}}(b)$ and $\mathcal{I}_{\text{ground}}(b)$ are also used for determining whether the bone b is touching the wall and the ground or not, respectively.

Climbing Stance: The assignment of the holds to the climber's hands/feet is called a climbing stance and denoted by $\sigma = \{\mathbf{x}_{ll}, \mathbf{x}_{rl}, \mathbf{x}_{lh}, \mathbf{x}_{rh}\}$. Each $\mathbf{x}_i \in \sigma$ can be set to be free $\mathbf{x}_{-1}$ or attached to a hold $\mathbf{x}_h \in \mathcal{H}$ on the wall. At the beginning of the training and/or climbing, the climber stands on the ground at T-pose where it is in stance $\sigma_0 = \{\mathbf{x}_{-1}, \mathbf{x}_{-1}, \mathbf{x}_{-1}, \mathbf{x}_{-1}\}$ and state $\mathbf{s}_0$.

B. Reinforcement Learning

In reinforcement learning, we have an agent in an environment, and the goal is for the agent to learn how to act in order to maximize the reward received from the environment. At each timestep t, the agent observes the current state $\mathbf{o}_t$, and chooses its next action $\mathbf{a}_t$ using its policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{o}_t)$, which is a mapping from states to a distribution over actions. After executing the action $\mathbf{a}_t$, the agent receives a scalar reward r_t, and observes the new state $\mathbf{o}_{t+1}$. The goal is to maximize the expected accumulated reward, i.e., $\mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$, where $\gamma \in [0,1]$ is the discount factor.

In deep reinforcement learning and continuous control, the policy is usually modeled using a neural network, whose parameters are denoted by θ. The policy network of $\pi_\theta(\mathbf{a}_t \mid \mathbf{o}_t)$ usually outputs the mean $\mu_\theta(\mathbf{a}_t \mid \mathbf{o}_t)$ and covariance $\mathcal{C}_\theta(\mathbf{a}_t \mid \mathbf{o}_t)$ to specify the observation-dependent distribution over actions. In this paper, we use ML-Agents [2], a Unity[1] plugin that provides an open-source[2] implementation of *Proximal Policy Optimization (PPO)* [10], a popular RL algorithm for continuous control.

IV. Method: Reinforcement Learning Framework for Climbing

This section introduces our reinforcement learning framework for synthesizing climbing movements. In Section IV-A, we formulate climbing movement synthesis as a reinforcement learning problem. In Sections IV-C and IV-B, we propose an exploration strategy that greatly affects the variety of learned movements. Finally, in Section IV-D, we explain how different action parameterizations can be used for smoothing the climbing movements. All parameter values are reported in Section V-A.

A. Reinforcement Learning Formulation

We now explain how learning humanoid climbing movements is formulated as a reinforcement learning problem.

[1]https://unity.com/

[2]https://github.com/Unity-Technologies/ml-agents

*1) **Agent's Observation**:* At each timestep t, The observation vector $\mathbf{o}_t$ is defined as follows:

$$\mathbf{o}_t = \langle \mathbf{s}_t, \sigma_c, \sigma_d, \mathcal{I}(\sigma_c), \mathcal{I}(\sigma_d), \mathcal{I}(\sigma_c \neq \sigma_d) \rangle \,, \tag{1}$$

where $\mathbf{s}_t$ and σ_c are the agent's current climbing state and stance (as defined in Section III). σ_d is the desired climbing stance for hands/feet positions set by the user or a high-level path planner while σ_c depends on the current state of the climber and shows its current hands/feet's positions. $\mathcal{I}(\sigma_c)$ and $\mathcal{I}(\sigma_d)$ indicate whether the hands/feet are attached to the holds in the current and desired stance, respectively. $\mathcal{I}(\sigma_c \neq \sigma_d)$ expresses which elements in the current and desired stances differ from each other.

*2) **Agent's Action**:* The sampled action vector $\mathbf{a}_t$ at timestep t has 44 values in the range $[-1,1]$. The first 30 elements of $\mathbf{a}_t$ denote the randomized target angles of climber's joints, and the last 14 elements are mapped to the range $[0, \tau_{\text{max}}]$ to be used as the joints' strength. Joint limits and τ_{max} are set such that the climber has human-like limitations.

*3) **Reward Function**:* Defining the reward r_t was one of the challenges in developing this framework. If a binary reward such as $r_t \in \{-1,1\}$ is used for only implying the success and failure of the climber, the learning is prohibitively slow due to the problem of sparse rewards. In order to speed up the learning process, the instantaneous reward function should be informative enough to guide the policy towards getting the hands/feet closer to the target stance at each timestep. Motivated by Deepmimic's reward function [3], we used the following distance-based reward function:

$$\begin{aligned} r_t^\sigma = &\mathcal{I}(d_t) \sum_{i \in \{\text{ll,rl,lh,rh}\}} \begin{bmatrix} k_\sigma \exp(-c_\sigma \|\sigma_{c,i} - \sigma_{d,i}\|) \\ + \mathcal{I}(\sigma_{c,i} = \sigma_{d,i}) \end{bmatrix}, \\ &-\mathcal{I}_{\text{ground}}(\mathbf{s}) \end{aligned} \tag{2}$$

where $\mathcal{I}(d_t)$ is an indicator function for specifying whether the agent has become closer to the target stance more than it was ever before. Without $\mathcal{I}(d_t)$ the agent oscillates near the target holds instead of attaching its hands or feet to them. Using $\mathcal{I}(d_t)$ yields more stable reaching behaviors. k_σ and c_σ are tuning parameters. $\mathcal{I}(\sigma_{c,i} = \sigma_{d,i})$ is 1 when climber's hands/feet reach their desired positions $\sigma_{d,i}$. Note that in principle, the reward should be a function of the observation and action. Thus, the data for computing $\mathcal{I}(d_t)$, i.e., previous closest achieved distance, should be added to the observation. However, the observation of Eq. 1 appears to work in practice.

Using the reward function shown in Eq. 2 does not guarantee that the policy produces smooth, energy conserving climbing movements. In order to encourage smooth movements, we incorporate a reward function based on the strengths of the joints, formulated as:

$$r_t^\tau = k_\tau \mathcal{I}(d_t) \exp\left[-c_\tau \sum_{i \in \{31,\ldots,44\}} (\mathbf{a}_t[i] + 1)^2\right], \tag{3}$$

where $\mathcal{I}(d_t)$ has the same functionality as in Eq. 2. The function sums over sampled strength values produced by the

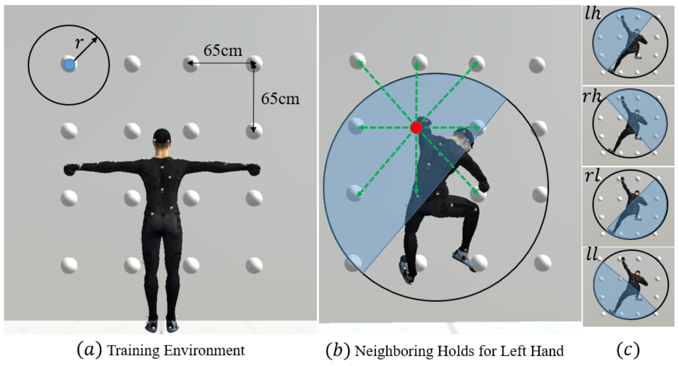

Fig. 2: (a) Initial hold positions with their randomization radius r. The climber is at T-pose in front of wall. (b) The neighboring holds for the climber's left hand. The neighbor holds include holds that are inside the shaded area, and connected by dashed lines to the hold that is shown by red dot. (c) The four shaded areas show the valid regions of the holds to be reached by the climber's hands/feet based on climber's hip position.

policy. c_τ and k_τ are tuning parameters. k_τ is lower than k_σ in Eq. 2 to make smoothness less important than the main objective of the policy is getting hands/feet to the target stance. Finally, r_t is calculated as the sum of Eq. 2 and Eq. 3, i.e.,

$$r_t = r_t^\sigma + r_t^\tau. \tag{4}$$

*4) **Training Episodes and Termination Condition**:* PPO assumes that experience is collected during training as episodes that start from some initial climbing state and run up to a time limit T_{episode} or a terminal climbing state. In our case, a climbing state is terminal if the climber reaches the target stance σ_d or any part of the climber's body except its feet touches the ground.

B. Exploration of Target Stances

The climber explores different climbing movements by sampling a target stance for each episode. However, not every transition is valid from a given episode's initial climbing state. A transition from a climbing state $\mathbf{s}$ and the stance σ, to the desired stance σ_d is valid if all of the following *Validity Conditions* are met:

- **Connectivity Limit:** At least one hand should be connected in both σ and σ_d, except for when the climber is at T-pose ($\sigma = \sigma_0$) and only σ_d needs to have at least one connected hand.
- **Distance Limit:** The distances between hand-to-hand, foot-to-foot, and hand-to-foot should not exceed the climber's body reach in both current and target stances.
- **Limb Movement Limit:** σ and σ_d should have at most two different elements.

In this paper, we use the following target stance sampling strategy. To sample a random target stance, we first randomly perturb all holds on the wall around the initial grid locations within $r = 33cm$ (Fig. 2-(a)). Then, we sample new target holds for 1 or 2 limbs among the neighboring holds of the moving limbs, as illustrated in Fig. 2-(b). The limb's neighboring holds includes the following:

- $\mathbf{x}_{-1}$, i.e., the target can be just freeing the limb.
- The holds inside of the shaded area for the limb (Fig. 2-(c)). The shaded area is within r_{body} around the current climber's hip position.
- The holds connected by dashed-lines to the hold attached to the moving limb.

The target holds for the moving limbs are selected uniformly from neighbour holds set.

C. Exploration of Climbing States

As shown by [3], the initial state distribution of training episodes can have a huge impact on learning complex humanoid movements. Their RSI technique initializes the agent to a random state sampled from reference motion capture data. In our case, the initial climbing state of an training episode has an important role in feasibility and exploration of the climbing moves. Promoting diverse exploration of climbing moves is conditioned to having rich initial climbing state distribution. We implement and compare the following two climbing state initialization approaches:

*1) **Standard episodic RL (Baseline)**:* Here, similar to common continuous control benchmarks like MuJoCo environments, we initialize the climber to the T-pose in front of a randomized wall, as illustrated in Fig 2, and climbing continues one target stance after another. We reset back to T-pose and randomize the wall after failure or a time limit (T_{baseline}). Failures are defined as reaching a terminal state (see Section IV-A4) without reaching the target stance. Note, that each episode of experience seen by PPO only includes one move to one desired stance, which makes value estimation much less noisy, as value does not depend on future desired stances. In initial testing, we found this to greatly improve performance.

Algorithm 1 Uniform Exploration on Seen Climbing States

```
1: for iteration = $1, 2, ..., \mathcal{I}_{\max}$ do
2:     $\mathcal{S} = \{\mathbf{s}_0\}$ //Initial climbing state distribution
3:     while Training budget $N_{\mathcal{D}}$ not exceeded do
4:         RandomizeScene
5:         Sample $\mathbf{s}$ uniformly from $\mathcal{S}$
6:         RandomRelocate $\mathbf{s}$ on the wall if all limbs connected, see Fig. 3
7:         Sample $\sigma_d$ from Section IV-B
8:         Run $\pi_\theta$ from $\mathbf{s}$ until episode ends at $\mathbf{s}_e$ (see Section IV-A4)
9:         $\mathcal{S} = \mathcal{S} \cup \{\mathbf{s}_e\}$ if either of the hands connected
10:    Update $\theta$ using PPO based on the observed experience tuples $[\mathbf{o}_t, \mathbf{a}_t, r_t, \mathbf{o}_{t+1}]$
```

*2) **Self-supervised episode state initialization (SS-ESI)**:* As detailed in Alg. 1, we gradually expand the set of possible initial climbing states $\mathcal{S}$ by collecting both failed and succeeded climbing moves (Line 9). In this case, each episode

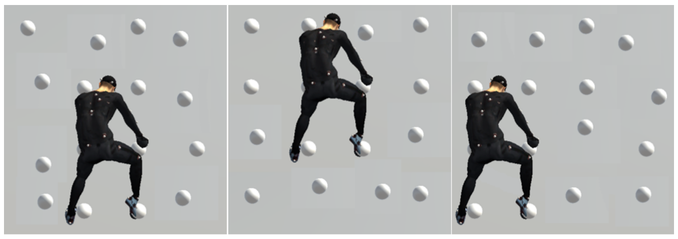

Fig. 3: Before selecting a new desired stance (Alg. 1 Line 7), the climber is randomly relocated on the climbing wall to allow more diverse movement. The positions of the holds not used by the climber are also randomly perturbed.

comprises only one climbing move that attempts to reach a single desired stance. In effect, the episode trajectories fork previous trajectories, forming an exploration tree. For each training iteration, we start in Line 2 in T-pose (s_0). Before each policy update, Lines 4-9 gradually gather more climbing states in $\mathcal{S}$. s_e is added to $\mathcal{S}$ if at least one hand is connected to a hold. In order to have more diverse and uniform climbing movement experience, we make sure in Alg. 1 that:

- The initial climbing state (s) of the training episodes is selected uniformly (Line 5) from seen/explored climbing states and relocated (Line 6) on the climbing wall, as illustrated in Fig. 3. We do this relocation only if all hands and feet are connected to avoid breaking the dynamics simulation.
- The sample distribution of target stances (Line 7) is balanced such that we sample a similar amounts of feet and hand moves.

D. *Action Parameterization*

We compare both single-step and multi-step actions. Multi-step actions simulate every action that comes from the policy for L steps instead of only 1 step. Compared to single-step action, using multi-step actions yields less experience tuples for the same amount of timesteps simulated.

V. Experiments

In this section we explain our experiments with training of the climber. We train the climber with 3 different settings:

1) Using 1-step action, and exploring the climbing movement by Alg. 1 (see SS-ESI in Section IV-C2).
2) Using 1-step action, and exploring the climbing movement by *Baseline* approach (see Section IV-C1).
3) Using 4-step action (see Section IV-D), and exploring the climbing movement by Alg. 1.

The first 2 training settings allow comparing the state initialization strategies of Section IV-C1 and IV-C2. The initialization strategy of Section IV-C2 (Alg. 1, SS-ESI) performed better; hence, we used it when testing the 4-step actions.

A. *Training Parameters*

Table I determines the parameter values used to train the climber with experiments. The first column establishes parameter values used for exploration strategies, action parameterization and designing of our reward function. The second column defines PPO's parameter values. $\mathcal{I}_{\text{max}}$, $N_{\mathcal{D}}$, and T_{episode} are maximum training iterations, experience budget, and maximum steps in training episode, respectively, that are used in Alg. 1. α is the initial learning rate value used in gradient decent updates of PPO. β is the PPO entropy loss weight, and ϵ is the clipped surrogate loss parameter. γ is the reward discount factor, and λ is the regularization parameter.

TABLE I: Parameters used during the training

Name	Value	Name	Value
r_{body}	130cm	$N_{\mathcal{D}}$	20480
τ_{max}	$250N.m$	T_{episode}	5 sec
c_σ	4	Minibatch size	2048
c_τ	3.6	Num. epoch	3
k_σ	$0.8/L$	β	$5.0e^{-3}$
k_τ	$0.2/L$	γ	0.995
T_{baseline}	50 sec	ϵ	0.2
$\mathcal{I}_{\text{max}}$	$1e^7$	λ	0.95
α	$1.0e^{-4}$		

Both the value and policy networks have 3 hidden layers with 256 units with `tanh` activation functions.

B. *Training Procedure and Curriculum*

In order to train the climber for moving more than one limb at a time, we increase the chance of sampling a target stance that requires 2-limb movement from 0% to 50% gradually as the training reaches half of max iterations.

During the training, we used a decaying learning rate where it started from the initial value specified in Table. I and decayed to $2.5e^{-7}$. However, we divided the training into 10 segments such that each segment comprises 10% of max training iterations and at the beginning of each segment the learning rate restarts at the initial value. This process helps to learn climbing movements better (see Fig. 4), the motivation for restarting the learning rate is to allow the policy optimization to reconverge as the optimization landscape gradually changes because of curriculum learning.

C. *Evaluation Settings*

All the training settings must be evaluated using the same distribution of climbing movements. We use Alg. 1 to collect $1k$ data evaluation episodes, i.e., succeeded or failed climbing moves, on trained models saved every $750k$ simulation steps during the training. This ensures a diverse and challenging evaluation distribution. In addition, k_σ and k_τ in Eq. 2 and 3 are set to 0.8 and 0.2 for evaluation of all experiments.

VI. Results

Here, we compare the training settings in terms of average success rate and cumulative reward. We train the models with their own exploration strategies and action parameterization,

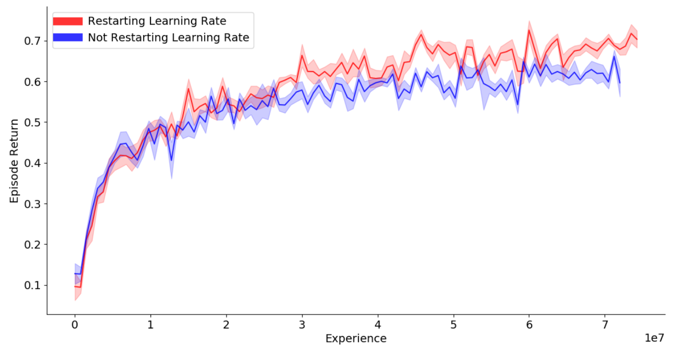

Fig. 4: The effect of restarting learning rate every $15M$ experience tuples. The motivation for this is to allow the policy optimization to reconverge as the optimization landscape gradually changes because of curriculum learning.

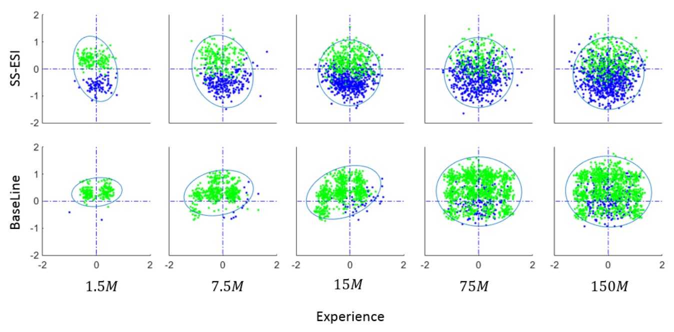

Fig. 5: The effect of our proposed Self-Supervised Episode State Initialization (SS-ESI) approach on the diversity of successful transitions during training. The green and blue dots show successful transitions of hands and feet, respectively, in the climbers local coordinates.

and gather $1k$ climbing movement samples for every model that is saved in every $750k$ of training iterations with the following evaluation settings. As the results depend somewhat on random seed, we repeat the experiments 5 times.

A. Visualizing the Exploration Methods

Fig. 5 illustrates how the episode initial state sampling affects exploration. Until $15M$ experience tuples, the policy fails a lot in performing successful climbing moves. With the baseline, where a failure resets the climbing to the T-pose, leg moves get almost no exploration until $15M$ experience tuples, and as the training continues, the learned policy explores an uneven distribution of the hand and foot movements. SS-ESI (Alg. 1) explores both hand and foot moves much more evenly during all training iterations.

B. Quantitative Results

Fig. 6 and Fig. 7 show the learning curves using both the baseline and the proposed self-supervised episode state initialization (SS-ESI) technique. As expected, the baseline learns much slower initially, although the difference in Fig. 6 becomes smaller as training progresses. Furthermore, Fig.

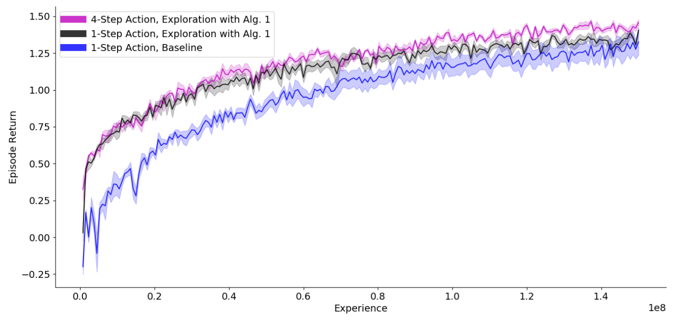

Fig. 6: Mean and standard deviation of episode return, from 5 independent training runs.

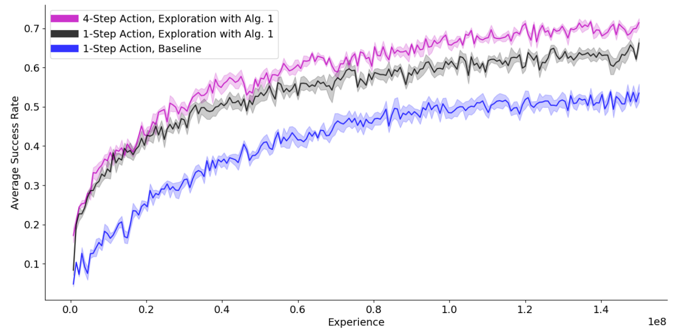

Fig. 7: Mean and standard deviation of episode success rate, from 5 independent training runs.

7 shows that the 1-step baseline approach has lower success rate than our approach in performing climbing movements. This difference does not vanish as training progresses. We may conclude that in complex and long movement sequences like climbing, SS-ESI can be helpful.

Furthermore, using multi-step action is the recommended choice, as the figures show that it leads to more rapid learning, and higher episode returns. As an additional benefit, the multi-step actions produce smoother, visually more pleasing movements. A visual inspection of the movements reveals that multi-step actions lead to wider movement arcs and thus wider exploration of climber states, which may explain the better performance.

On the other hand, increasing the number of simulation steps per action makes collecting experience more costly. If the step count is increased too much, it also limits the complexity and temporal resolution of movements. However, our experiments show that a moderate increase of steps per action can be beneficial, compared to the baseline of taking a new action at every simulation step.

VII. Conclusions, Limitations and Future Work

We have formulated the problem of simulated humanoid climbing as a reinforcement learning problem. Our experiments show that 1) our proposed self-supervised episode state initialization approach improves exploration and speeds up learning, and 2) using multi-step actions likewise improves learning and also leads to visually smoother and more realistic movements. As a result, we can produce an efficient neural network policy for interactive physically simulated climbing. This has applications in both games and cognitive practice of real-life climbing [25], and the episode state initialization approach may be useful in other movement problems where long sequences of complex movements need to be trained without reference data.

A primary limitation that will be addressed in future work is that we do not presently model the finger and climbing hold details. We have also limited our experiments to moving only 1 or 2 limbs at the same time. While this can model a large variety of real-life climbing movements, simulating climbing on a high skill level also needs to handle dynamic full-body leaps. We believe our approach can be extended by introducing leaping movements to the training curriculum, although this will likely need much more simulated experience and/or a more efficient policy optimization method than PPO. Good candidates for this are PPO-CMA [12], SAC [5], or relative entropy regularized policy iteration [26].

Acknowledgment

This work has been supported by Academy of Finland, grants 299358 and 305737.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] A. Juliani, V.-P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," *arXiv preprint arXiv:1809.02627*, 2018.

[3] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 143, 2018.

[4] L. Liu and J. Hodgins, "Learning to schedule control fragments for physics-based characters using deep q-learning," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 3, p. 29, 2017.

[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[7] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[9] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization." in *Icml*, vol. 37, 2015, pp. 1889–1897.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[11] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[12] P. Hämäläinen, A. Babadi, X. Ma, and J. Lehtinen, "PPO-CMA: Proximal policy optimization with covariance matrix adaptation," *arXiv preprint arXiv:1810.02541*, 2018.

[13] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller, "Maximum a posteriori policy optimisation," *arXiv preprint arXiv:1806.06920*, 2018.

[14] A. Abdolmaleki, J. T. Springenberg, J. Degrave, S. Bohez, Y. Tassa, D. Belov, N. Heess, and M. Riedmiller, "Relative entropy regularized policy iteration," *arXiv preprint arXiv:1812.02256*, 2018.

[15] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq *et al.*, "Deepmind control suite," *arXiv preprint arXiv:1801.00690*, 2018.

[16] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[17] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 317–342, 2006.

[18] B. Libeau, A. Micaelli, and O. Sigaud, "Transfer of knowledge for a climbing virtual human: A reinforcement learning approach," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2119–2124.

[19] S. Jain, Y. Ye, and C. K. Liu, "Optimization-based interactive motion synthesis," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 1, p. 10, 2009.

[20] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 43, 2012.

[21] K. Naderi, J. Rajamäki, and P. Hämäläinen, "Discovering and synthesizing humanoid climbing movements," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 43:1–43:11, Jul. 2017. [Online]. Available: http://doi.acm.org/10.1145/3072959.3073707

[22] K. Naderi, A. Babadi, and P. Hämäläinen, "Learning physically based humanoid climbing movements," in *Computer Graphics Forum*, vol. 37, no. 8. Wiley Online Library, 2018, pp. 69–80.

[23] T. Olsen, S. Andrews, and P. Kry, "Computational Climbing for Physics-Based Characters," *The ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA posters)*, 2014.

[24] S. Tonneau, P. Fernbach, A. D. Prete, J. Pettré, and N. Mansard, "2pac: Two-point attractors for center of mass trajectories in multi-contact scenarios," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 5, p. 176, 2018.

[25] K. Naderi, J. Takatalo, J. Lipsanen, and P. Hämäläinen, "Computer-aided imagery in sport and exercise: A case study of indoor wall climbing," in *Proceedings of Graphics Interface 2018*, ser. GI 2018. Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, 2018, pp. 93 – 99.

[26] A. Abdolmaleki, J. T. Springenberg, J. Degrave, S. Bohez, Y. Tassa, D. Belov, N. Heess, and M. A. Riedmiller, "Relative entropy regularized policy iteration," *CoRR*, vol. abs/1812.02256, 2018. [Online]. Available: http://arxiv.org/abs/1812.02256