Fig. 1. Simulated characters performing highly dynamic skills learned by imitating video clips of human demonstrations. **Left:** Humanoid performing cartwheel B on irregular terrain. **Right:** Backflip A retargeted to a simulated Atlas robot.

Data-driven character animation based on motion capture can produce highly naturalistic behaviors and, when combined with physics simulation, can provide for natural procedural responses to physical perturbations, environmental changes, and morphological discrepancies. Motion capture remains the most popular source of motion data, but collecting mocap data typically requires heavily instrumented environments and actors. In this paper, we propose a method that enables physically simulated characters to learn skills from videos (SFV). Our approach, based on deep pose estimation and deep reinforcement learning, allows data-driven animation to leverage the abundance of publicly available video clips from the web, such as those from YouTube. This has the potential to enable fast and easy design of character controllers simply by querying for video recordings of the desired behavior. The resulting controllers are robust to perturbations, can be adapted to new settings, can perform basic object interactions, and can be retargeted to new morphologies via reinforcement learning. We further demonstrate that our method can predict potential human motions from still images, by forward simulation of learned controllers initialized from the observed pose. Our framework is able to learn a broad range of dynamic skills, including locomotion, acrobatics, and martial arts. (Video1)

1 INTRODUCTION

Data-driven methods have been a cornerstone of character animation for decades, with motion-capture being one of the most popular sources of motion data. Mocap data is a staple for kinematic methods, and is also widely used in physics-based character animation. Imitation of mocap clips has been shown to be an effective approach for developing controllers for simulated characters, yielding some of the most diverse and naturalistic behaviors. However, the acquisition of mocap data can pose major hurdles for practitioners, often requiring heavily instrumented environments and actors. The infrastructure required to procure such data can be prohibitive, and some activities remain exceedingly difficult to motion capture, such as large-scale outdoor sports. A more abundant and flexible source of motion data is monocular video. A staggering 300 hours of video is uploaded to YouTube every minute [Aslam 2018]. Searching and querying video sources on the web can quickly yield a large number of clips for any desired activity or behavior. However, it is a daunting challenge to extract the necessary motion information from monocular video frames, and the quality of the motions generated by previous methods still falls well behind the best mocap-based animation systems [Vondrak et al. 2012].
In this paper, we propose a method for acquiring dynamic character controllers directly from monocular video through a combination of pose estimation and deep reinforcement learning. Recent advances with deep learning techniques have produced breakthrough results for vision-based 3D pose estimation from monocular images [Kanazawa et al. 2018]. However, pose estimation alone is not yet sufficient to produce high-fidelity and physically plausible motions: frequent errors and physical inconsistencies in the estimated poses accumulate and result in unnatural character behaviors. Motion imitation with reinforcement learning provides a powerful tool for acquiring skills from videos while remaining robust to such imperfections. By reproducing the skill in a physical simulation, the learning process can refine imperfect and noisy pose sequences, compensate for missing frames, and take into account the physical constraints of the character and environment. By bringing together deep pose estimation and reinforcement learning, we propose a framework that enables simulated characters to learn a diverse collection of dynamic and acrobatic skills directly from video demonstrations.

The primary contribution of our paper is a system for learning character controllers from video clips that integrates pose estimation and reinforcement learning. To make this possible, we introduce a number of extensions to both the pose tracking system and the reinforcement learning algorithm. We propose a motion reconstruction method that improves the quality of reference motions to be more amenable for imitation by a simulated character. We further introduce a novel reinforcement learning method that incorporates an adaptive state initialization, where the initial state distribution is dynamically updated to facilitate long-horizon performance in reproducing a desired motion. We find that this approach for dynamic curriculum generation substantially outperforms standard methods when learning from low-fidelity reference motions constructed from video tracking sequences. Our framework is able to reproduce a significantly larger repertoire of skills and higher fidelity motions from videos than has been demonstrated by prior methods. The effectiveness of our framework is evaluated on a large set of challenging skills including dances, acrobatics, and martial arts. Our system is also able to retarget video demonstrations to widely different morphologies and environments. Figure 1 illustrates examples of the skills learned by our framework. Furthermore, we demonstrate a novel physics-based motion completion application that leverages a corpus of learned controllers to predict an actor’s full-body motion given a single still image. While our framework is able to reproduce a substantially larger corpus of skills than previous methods, there remains a large variety of video clips that our system is not yet able to imitate. We include a discussion of these challenges and other limitations that arise from the various design decisions.

2 RELATED WORK

Our work lies at the intersection of pose estimation and physics-based character animation. The end goal of our system is to produce robust and naturalistic controllers that enable virtual characters to perform complex skills in physically simulated environments. Facets of this problem have been studied in a large body of prior work, from techniques that have sought to produce realistic skills from first principles (i.e., physics and biomechanics) [Coros et al. 2010; Wampler et al. 2014; Wang et al. 2012], to methods that incorporate reference motion data into the controller construction process [da Silva et al. 2008; Lee et al. 2010a; Liu et al. 2010]. These techniques can synthesize motions kinematically [Holden et al. 2017; Lee et al. 2010b; Levine et al. 2012] or as the product of dynamic control in a physics simulation [Geijtenbeek et al. 2013; Lee et al. 2014]. Most data-driven methods, save for a few exceptions, are based on motion capture data, which often requires costly instrumentation and pre-processing [Holden et al. 2016]. Raw video offers a potentially more accessible and abundant alternative source of motion data. While there has been much progress in the computer vision community in predicting human poses from monocular images or videos, integrating pose predictions from video with data-driven character animation still presents a number of challenges. Pose estimators can generally produce reasonable predictions of an actor’s motion, but they do not benefit from the manual cleanup and accurate tracking enjoyed by professionally recorded mocap data. Prior methods that learn from motion data often assume accurate reference motions as a vital component in the learning process. For example, during training, [Peng et al. 2018] reinitializes the character state to frames sampled from the reference motion. The effectiveness of these strategies tend to deteriorate in the presence of low-fidelity reference motions.

Reinforcement Learning: Many methods for acquiring character controllers utilize reinforcement learning [Coros et al. 2009; Lee et al. 2010b; Levine et al. 2012; Peng et al. 2015; Wang et al. 2010]. The use of deep neural network models for RL has been demonstrated for a diverse array of challenging skills [Brockman et al. 2016; Duan et al. 2016; Liu and Hodgins 2017; Peng et al. 2016; Rajeswaran et al. 2017; Teh et al. 2017]. While deep RL methods have been effective for motion control tasks, the policies are prone to developing unnatural behaviours, such as awkward postures, overly energetic movements, and asymmetric gaits [Merel et al. 2017; Schulman et al. 2017]. In order to mitigate these artifacts, additional auxiliary objectives such as symmetry, effort minimization, or impact penalties have been incorporated into the objective to discourage unnatural behaviors [Yu et al. 2018b]. Designing effective objectives can require substantial human insight and may nonetheless fall short of eliminating undesirable behaviors. An alternative for encouraging more natural motions is to incorporate high-fidelity biomechanical models [Geijtenbeek et al. 2013; Lee et al. 2014; Wang et al. 2012]. However, these models can be challenging to build, difficult to control, and may still result in unnatural behaviors. In light of these challenges, data-driven RL methods that utilize reference motion data have been proposed as an alternative [Peng et al. 2018; Won et al. 2017]. Reference motion clips can be incorporated via a motion imitation objective that incentivizes the policy to produce behaviours that resemble the reference motions. In this paper, we explore methods for extending motion imitation with RL to accommodate low-fidelity reference motions extracted from videos, and introduce a novel adaptive state initialization technique that makes this practical even for highly dynamic and acrobatic movements.
Monocular Human Pose Estimation: While mocap remains the most popular source of demonstrations, it typically requires significant instrumentation, which limits its accessibility. Practitioners therefore often turn to public databases to satisfy their mocap needs [CMU 2018; SFU 2018]. Unfortunately, the volume of publicly available mocap data is severely limited compared to datasets in other disciplines, such as ImageNet [Deng et al. 2009]. Alternatively, video clips are an abundant and accessible source of motion data. While recovering motion from raw video has been a long standing challenge [Bregler and Malik 1998; Lee and Chen 1985], recently deep learning approaches have made rapid progress in this area.

Performance of 2D pose estimation improved rapidly after Toshev and Szegedy [2014] introduced a deep learning approach for predicting the 2D coordinates of joints directly from images. This is followed by methods that predict joint locations as a spatial heat map [Newell et al. 2016; Tompson et al. 2014; Wei et al. 2016]. In this work we build upon the recent OpenPose framework [Cao et al. 2017], which extends previous methods for real-time multi-person 2D pose estimation. Monocular 3D pose estimation is an even more challenging problem due to depth ambiguity, which traditional methods resolve with strong priors [Bogo et al. 2016; Taylor 2000; Zhou et al. 2015]. The introduction of large-scale mocap datasets [Ionescu et al. 2014] with ground truth 3D joint locations allowed for the development of deep learning based methods that directly estimate 3D joint locations from images [Mehta et al. 2017; Pavlakos et al. 2017; Zhou et al. 2016]. However, mocap datasets are typically captured in heavily instrumented environments, and models trained on these datasets alone do not generalize well to the complexity of images of humans in the wild. Therefore, recent methods focus on weakly supervised techniques, where a model may also be trained on images without ground truth 3D pose [Rogez and Schmid 2016; Zhou et al. 2017]. Note that most approaches only estimate the 3D joint locations and not the 3D rotations of a kinematic tree, which is necessary to serve as reference for our RL algorithm. Methods that predict joint locations require additional post-processing to recover the joint rotations through inverse kinematics [Mehta et al. 2017]. Only a handful of techniques directly estimate the 3D human pose as 3D joint rotations [Kanazawa et al. 2018; Tung et al. 2017; Zhou et al. 2016]. Although there are methods that utilize video sequences as input [Tekin et al. 2016], most state-of-the-art approaches predict the pose independently for each video frame. Recently Xu et al. [2018] propose a method that recovers a temporally consistent trajectory from monocular video by an additional optimization step in the 3D pose space. However, their method requires a pre-acquired template mesh of the actor and hence cannot be applied to legacy videos, such as those available from YouTube. In this work we build on the recent work of Kanazawa et al. [2018], which is a weakly-supervised deep learning framework that trains a model to directly predict the 3D pose, as joint rotations, from a single image. A more detailed discussion is available in Section 4.

Video Imitation: The problem of learning controllers from monocular video has received modest attention from the computer graphics community. The work most related to ours is the previous effort by Vondrak et al. [2012], which demonstrated learning bipedal controllers for walking, jumping, and handsprings from videos. The controllers were represented as a finite-state machines (FSM), where the structure of the FSM and the parameters at each state were learned through an incremental optimization process. Manually-crafted balance strategies and inverse-dynamics models were incorporated into the control structure within each state of the FSM. To imitate the motion of the actor in a video, the controllers were trained by optimizing a 2D silhouette likelihood computed between the actor and simulated character. To resolve depth ambiguity, they incorporated a task-specific pose prior computed from mocap data. While the system was able to synthesize controllers for a number of skills from video demonstrations, the resulting motions can appear robotic and the use of a silhouette likelihood can neglect a significant amount of task-relevant information in the video. Furthermore, the task-pose priors require access to mocap clips that are similar to the skills being learned. If such data is already available, it might be advantageous to imitate the mocap clips instead. Similarly, Coros et al. [2011] utilized video clips of canine motions to train quadraped controllers, where the reference motions were extracted via manually annotating gait graphs and marker locations.

In this work, we take advantage of state-of-the-art 3D pose estimation techniques to extract full-body 3D reference motions from video, which resolves much of the depth ambiguity inherent in monocular images and improves the motion quality of the learned controllers. Deep RL enables the use of simple but general control structures that can be applied to a substantially wider range of skills, including locomotion, acrobatics, martial arts, and dancing. Our approach can be further extended to a novel physics-based motion completion application, where plausible future motions of a human actor can be predicted from a single image by leveraging a library of learned controllers. While our framework combines several components proposed in prior work, including the use of vision-based pose estimators [Kanazawa et al. 2018; Wei et al. 2016] and deep reinforcement learning with reference motion data [Peng et al. 2018], the particular combination of these components is novel, and we introduce a number of extensions that are critical for integrating these disparate systems. To the best of our knowledge, the only prior work that has demonstrated learning full-body controllers from monocular video is the work by Vondrak et al. [2012]. Although incorporating reinforcement learning to imitate video demonstrations is conceptually natural, in practice it presents a number of challenges arising from nonphysical behaviours and other artifacts due to inaccurate pose estimation.

3 OVERVIEW

Our framework receives as input a video clip and a simulated character model. It then synthesizes a controller that enables a physically simulated character to perform the skill demonstrated by the actor in the video. The resulting policies are robust to significant perturbations, can be retargeted to different characters and environments, and are usable in interactive settings. The learning process is divided into three stages: pose estimation, motion reconstruction, and motion imitation. A schematic illustration of the framework is available in Figure 2. The input video is first processed by the
which is vital for learning acrobatic skills from video clips.

ACM Transactions on Graphics, Vol. 37, No. 6, Article 178. Publication date: November 2018.

178:4

• Lin et al. 2014, and sports datasets Johnson and Everingham 2010,

2D pose annotations, framework that allows the model to be trained on images with only

to produce the pose. HMR uses a weakly-supervised adversarial

pose estimation stage, where a learned 2D and 3D pose estimators are applied to extract the pose of the actor in each frame. Next, the set of predicted poses proceeds to the motion reconstruction stage, where a reference motion trajectory \( \{q_t^r\} \) is optimized such that it is consistent with both the 2D and 3D pose predictions, while also enforcing temporal-consistency between frames and mitigating other artifacts present in the original set of predicted poses. The reference motion is then utilized in the motion imitation stage, where a control policy \( \pi \) is trained to enable the character to reproduce the reference motion in a physically simulated environment. The pose estimator is trained with a weakly-supervised learning approach, and the control policy is trained with reinforcement learning using a motion imitation objective.

4 BACKGROUND

Pose estimation: Our approach builds upon the recent 2D and 3D pose estimators, OpenPose [Wei et al. 2016] and Human Mesh Recovery (HMR) [Kanazawa et al. 2018] respectively. OpenPose performs both detection and 2D pose estimation of humans from a single image. It outputs the 2D pose as joint locations \( x_j \in \mathbb{R}^2 \) in the image coordinate space, as well as a confidence score for each joint \( c_j \in \mathbb{R} \). HMR is a recent approach that directly predicts the 3D pose and shape of a human model [Loper et al. 2015], along with the camera configuration from an image of a localized person. The predicted 3D pose \( q = \{q_j\} \) is parameterized by the local rotation of each joint \( q_j \), represented in axis-angle form with respect to the parent link’s coordinate frame. The world transformation of the root, designated to be the pelvis, is obtained using the predicted weak-perspective camera \( \Pi \). The 3D pose is predicted by first encoding an image \( I \) into a 2048D latent space \( z = f(I) \) via a learned encoder \( f \). The latent features are then decoded by a learned decoder \( q(z) \) to produce the pose. HMR uses a weakly-supervised adversarial framework that allows the model to be trained on images with only 2D pose annotations, without any ground truth 3D labels. Therefore, it can be trained on datasets of in-the-wild images, such as COCO [Lin et al. 2014], and sports datasets Johnson and Everingham 2010], which is vital for learning acrobatic skills from video clips.

Reinforcement Learning: Our algorithm makes use of reinforcement learning, which has previously been used for imitation of mocap data [Liu et al. 2016; Feng et al. 2018]. During the motion imitation stage, the control policy is trained to imitate a reference motion via a motion imitation objective. Training proceeds by having an agent interact with its environment according to a policy \( \pi(a|s) \), which models the conditional distribution of action \( a \) given a state \( s \in S \). At each timestep \( t \), the agent observes the current state \( s_t \) and samples an action \( a_t \) from \( \pi \). The environment then responds with a successor state \( s_t' = s_{t+1} \), sampled from the dynamics \( p(s_{t+1}|s_t, a_t) \), and a scalar reward \( r_t \), which reflects the desirability of the transition. For a parametric policy \( \pi_\theta(a|s) \), with parameters \( \theta \), the goal of the agent is to learn the optimal parameters \( \theta^* \) that maximizes its expected return

\[
J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[ \sum_{t=0}^{T-1} \gamma^t r_t \right],
\]

where \( p_\theta(\tau) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t) \) is the distribution over trajectories \( \tau = (s_0, a_0, s_1, \ldots, a_{T-1}, s_T) \) induced by the policy \( \pi_\theta \), with \( p(s_0) \) being the initial state distribution. \( \sum_{t=0}^{T} \gamma^t r_t \) represents the discounted return of a trajectory, with a horizon of \( T \) steps and a discount factor \( \gamma \in [0, 1] \).

Policy gradient methods are a popular class of algorithms for optimizing parametric policies [Sutton et al. 2001]. The algorithm optimizes \( J(\theta) \) via gradient ascent, where the gradient of the objective with respect to the policy parameters \( \nabla_\theta J(\theta) \) is estimated using trajectories that are obtained by rolling out the policy:

\[
\nabla_\theta J(\theta) = \mathbb{E}_{s_t \sim d_\theta(s_t), a_t \sim \pi_\theta(a_t|s_t)} \left[ \nabla_\theta \log(\pi_\theta(a_t|s_t)) A_t \right],
\]

where \( d_\theta(s_t) \) is the state distribution under the policy \( \pi_\theta \). \( A_t = R_t - V(s_t) \) represents the advantage of taking an action \( a_t \) at a given state \( s_t \), with \( R_t = \sum_{t=0}^{T-1} \gamma^t r_{t+1} \) being the return received by a particular trajectory starting from state \( s_t \) at time \( t \) and \( V(s_t) \) is a value function that estimates the average return of starting in \( s_t \) and following the policy for all subsequent steps. A number of practical improvements have been proposed, such as trust regions [Schulman et al. 2015a], natural gradient [Kakade 2001], and entropy regularization to prevent premature distribution collapse [J. Williams and Peng 1991].

5 POSE ESTIMATION

Given a video clip, the role of the pose estimation stage is to predict the pose of the actor in each frame. Towards this goal, there are two main challenges for our task. First, the acrobatic skills that we wish to imitate exhibit challenging poses that vary significantly from the distribution of common poses available in most datasets. Second, poses are predicted independently for each frame, and therefore may not be temporally consistent, especially for dynamic motions. We address these challenges by leveraging an ensemble of pose estimators and a simple but effective data augmentation technique that substantially improve the quality of the predictions.

One of the challenges of tracking acrobatic movements is that they tend to exhibit complex poses with wildly varying body orientations (e.g. flips and spins). These poses are typically underrepresented in existing datasets, which are dominated by everyday images of...
With this augmented dataset, substantially improves performance. on the reference motion is more amenable for imitation. The predictions across adjacent frames (see supplementary video). The 3D consistency loss $l_{3D}$ encourages the optimized trajectory to stay close to the initial 3D prediction $q_t$: $l_{3D} = \sum_t w_t \text{dist}(q_t, q(z_t)),$ where $\text{dist}(\cdot, \cdot)$ measures the distance between two rotations by the angle of the difference rotation. $w_t = \exp(-\delta_t)$ estimates the confidence of the initial 3D prediction using the difference between the initial 2D and 3D predictions, computed via the reprojection error $\delta_t = \sum_j c_{t,j} |(\hat{x}_{t,j} - \Pi F_j(q(z_t)))|_2.$ This ensures that initial 3D poses that are consistent with the 2D predictions are preserved, while inconsistent poses are adjusted through the other terms in the loss. Finally, the smoothness loss $l_{sm}$ encourages smoothness of the 3D joint positions between adjacent frames $l_{sm} = \sum_t \sum_j \left\| F_j(q(z_t)) - F_j(q(z_{t+1})) \right\|_2^2.$ After the optimization process, we obtain the final 3D reference motion $\{q_t^*\} = \{q(z_t^*)\}$. 7 MOTION IMITATION WITH RL Once the reference motion has been reconstructed, it progresses to the motion imitation stage, where the goal is to learn a policy $\pi$ that enables the character to reproduce the demonstrated skill in simulation. The reference motion extracted by the previous stages is used to define an imitation objective, and a policy is then trained through reinforcement learning to imitate the given motion. The policy is modeled as a feedforward network that receives as input the current state $s$ and outputs an action distribution $\pi(a|s)$. To train the policy, we propose a variant of proximal policy optimization (PPO) [Schulman et al. 2017] augmented with an adaptive initial state distribution, as described below.
7.1 Initial State Distribution

The initial state distribution $p(s_0)$ determines the states at which an agent starts each episode. Careful choice of $p(s_0)$ can have a significant impact on the performance of the resulting policy, as it can mitigate the challenges of exploration inherent in RL. An effective initial state distribution should expose the agent to promising states that are likely to maximize its expected return, thereby reducing the need for the agent to discover such states on its own. In the case of learning to imitate a reference motion, sampling initial states from the target trajectory can be highly effective for reproducing dynamic motions, such as flips and spins [Peng et al. 2018]. But the effectiveness of this strategy depends heavily on the quality of the reference motion. Due to artifacts from the pose estimator and modeling discrepancies between the simulated character and real-world actor, states sampled from the reference motion may not be ideal for reproducing the entirety of the motion. For example, a common artifact present in motion data recorded from real-world actors, be it through motion capture or vision-based pose estimation, is high-frequency jittering, which can manifest as initial states with large joint velocities. Naively initializing the agent to such states will then require the agent to recover from the artifacts of the reference motion. These artifacts can be substantially more pronounced in reference motions reconstructed from video. Though the motion reconstruction stage is able to mitigate many of these artifacts, some errors may still persist.

While a myriad of post-processing techniques can be applied to mitigate artifacts in the reference motion, we can instead reduce the dependency on the quality of the reference motion by learning an initial state distribution with the specific purpose of aiding the character in learning an effective controller. This can be formulated as a cooperative multi-agent reinforcement learning problem where the first agent, defined by the policy $\pi(\theta, s)$, controls the movement of the character, and the second agent $\rho(\omega, s_0)$ proposes the initial states at which the character should start each episode. Both agents cooperate in order to maximize the multi-agent objective:

$$J(\theta, \omega) = \mathbb{E}_{T \sim p_{\theta,\omega}(\tau)} \left[ \sum_{t=0}^{T} y^t r_t \right]$$

$$= \int p(\omega) \prod_{t=0}^{T-1} p(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t) \sum_{t=0}^{T} y^t r_t \, dt.$$  

Note that, since the reward requires tracking the entire reference motion (as discussed in the next section), the initial state distribution cannot obtain the maximum return by "cheating" and providing excessively easy initializations. The maximum return is attained when the initial state distribution is close to the reference trajectory, but does not initialize the character in states from which recovery is impossible, as might be the case with erroneous states due to tracking error. The policy gradient of the initial state distribution $\rho(\omega, s_0)$ can be estimated according to:

$$= \mathbb{E}_{T \sim p_{\theta,\omega}(\tau)} \left[ \nabla_{\omega} \log p(\omega) \sum_{t=0}^{T} y^t r_t \right].$$

A detailed derivation is available in the supplementary material. Similar to the standard policy gradient for $\pi$, the gradient of the initial state distribution can be interpreted as increasing the likelihood of initial states that result in high returns. Unlike the standard policy gradient, which is calculated at every timestep, the gradient of the initial state distribution is calculated only at the first timestep. The discount factor captures the intuition that the effects of the initial state attenuates as the episode progresses. We will refer to this strategy of learning the initial state distribution as adaptive state initialization (ASI). Learning an initial state distribution can also be interpreted as a form of automatic curriculum generation, since $\rho(\omega, s_0)$ is incentivized to propose states that enable the character to closely reproduce the reference motion while also avoiding states that may be too challenging for the current policy.

7.2 Reward

The reward function is designed to encourage the character to match the reference motion ($q^*_t$) generated by the motion reconstruction stage. The reward function is similar to the imitation objective proposed by Peng et al. [2018], where at each step, the reward $r_t$ is calculated according to:

$$r_t = w^p r^p_t + w^v r^v_t + w^e r^e_t + w^w r^w_t$$

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

The pose reward $r^p_t$ incentivizes the character to track the joint orientations from the reference motion, computed by quaternion differences of the simulated character’s joint rotations and those of the reference motion. In the equation below, $q_{t,j}$ and $\dot{q}_{t,j}$ represent the rotation of the $j$th joint from the simulated character and reference motion respectively, $q_1 \oplus q_2$ denotes the quaternion difference, and $||q||$ computes the scalar rotation of a quaternion about its axis:

$$r^p_t = \exp \left[ -2 \left( \sum_j ||q^*_t \oplus q_{t,j}||^2 \right) \right].$$

Similarly, the velocity reward $r^v_t$ is calculated from the difference of local joint velocities, with $\dot{q}_{t,j}$ being the angular velocity of the $j$th joint. The target velocity $\dot{q}^t_{t,j}$ is computed from the reference motion via finite difference:

$$r^v_t = \exp \left[ -0.1 \left( \sum_j ||\dot{q}^t_{t,j} - \dot{q}_{t,j}||^2 \right) \right].$$

The end-effector reward $r^e_t$ encourages the character’s hands and feet to match the positions specified by the reference motion. Here, $p_{t,e}$ denotes the 3D position with respect to the root of end-effector $e \in \{\text{left foot, right foot, left hand, right hand}\}$:

$$r^e_t = \exp \left[ -40 \left( \sum_e ||p^*_t e - p_{t,e}||^2 \right) \right].$$

Finally, $r^w_t$ penalizes deviations in the character’s center-of-mass $c_t$ from that of the reference motion $c^*_t$:

$$r^w_t = \exp \left[ -10 \left( ||c^*_t - c_t||^2 \right) \right].$$

The positional qualities $p_{t,e}$ and $c^*_t$ are computed from the reference pose $q^*_t$ via forward kinematics.
7.3 Training

Training proceeds episodically, where at the start of each episode, the character is initialized to a state $s_0$ sampled from the initial state distribution $p_{0s}(s_0)$. A rollout is then simulated by sampling actions from the policy $\pi_0(a_t|s_t)$ at every step. An episode is simulated to a fixed time horizon or until a termination criteria has been triggered. Early termination is triggered whenever the character falls, which is detected as any link in the character’s torso coming into contact with the ground. Once terminated, the policy receives 0 reward for all remaining timesteps in the episode. This termination criteria is disabled for contact-rich motions, such as rolling. Once a batch of data has been collected, minibatches are sampled from the dataset and used to update the policy and value networks. TD($\lambda$) is used to compute target values for the value network $V_\theta(s)$ [Sutton and Barto 1998] and GAE($\lambda$) is used to compute advantages for policy updates [Schulman et al. 2015b]. Please refer to the supplementary material for a more detailed summary of the learning algorithm.

8 MOTION COMPLETION

Once a collection of policies has been trained for a corpus of different skills, we can leverage these policies along with a physics-based simulation to predict the future motions of actors in new scenarios. Given a single still image of a human actor, the goal of motion completion is to predict the actor’s motion in subsequent timesteps. We denote the library of skills as $\{\{q_i^t, \pi^i\}\}$, where $\pi^i$ is the policy trained to imitate reference motion $q_i^t$. To predict the actor’s motion, the target image is first processed by the 3D pose estimator to extract the pose of the actor $\dot{q}$. The extracted pose is then compared to every frame of each reference motion to select the reference motion and corresponding frame $q_i^\tau$, that is most similar to $\dot{q}$:

$$q_i^\tau = \arg \min_{q_i^\tau} ||q \otimes q_i^\tau||.$$  

Next, the simulated character is initialized to the state defined by the pose $\dot{q}$ and, since still images do not provide any velocity information, we initialize the character’s velocity to that of the selected reference motion $\dot{q}_i^\tau$. The motion is then simulated by applying the policy $\pi^i$ corresponding to the selected motion.

9 EXPERIMENTAL SETUP

Our framework will be demonstrated using a 3D humanoid character and a simulated Atlas robot. The humanoid is modeled as articulated rigid bodies with a total of 12 joints, a mass of 45kg, and a height of 1.62m. Each link in the character’s body is connected to its parent via a 3 degree-of-freedom spherical joint, except for the elbows and knees, which are modeled using a 1 degree-of-freedom revolute joint.

The simulated Atlas robot follows a similar body structure, with a mass of 169.5kg and a height of of 1.82m. The characters are actuated by PD controllers positioned at each joint, with manually specified gains and torque limits. Physics simulation is performed using the Bullet physics engine at 1.2kHz [Bullet 2015], with the policy being queried at 30Hz. All neural networks are built and trained using TensorFlow. Every episode is simulated for a maximum horizon of 20s. Each policy is trained with approximately 150 million samples, taking about 1 day on a 16-core machine.

Policy Network: The policy network $\pi(a|s)$ is constructed using 2 fully-connected layers with 1024 and 512 units respectively, with ReLU activations, followed by a linear output layer. The action distribution is modeled as a Gaussian with a state-dependent mean $\mu(s)$ and a fixed diagonal covariance matrix $\Sigma$:

$$\pi(a|s) = \mathcal{N}(\mu(s), \Sigma).$$

The value function $V_\theta(s)$ is represented by a similar network, with exception of the output layer, which produces a single scalar value.

State: The state $s$ consists of features that describe the configuration of the character’s body. The features are identical to those used by Peng et al. [2018], which include the relative positions of each link with respect to the root, their rotations represented by quaternions, and their linear and angular velocities. All features are computed in the character’s local coordinate frame, where the origin is located at the root and the x-axis pointing along the root link’s facing direction. Since the target pose from the reference motion varies with time, a scalar phase variable $\phi \in [0, 1]$ is included among the state features. $\phi = 0$ denotes the start of the motion, and $\phi = 1$ denotes the end.
denotes the end. The phase variables therefore helps to ensure that the policy is synchronized with the reference motion. Combined, the features constitute a 197D state space.

Action: The action s specifies target rotations for PD controllers positioned at each joint. For spherical joints, the targets are specified in a 4D axis-angle form. For revolute joints, the targets are specified by scalar rotation angles. Combined, the parameters result in a 36D action space.

Initial State Distribution: At the start of each episode, the character is initialized to a state s0 sampled from the initial state distribution \( \rho_{\omega}(s_0) \). When applying adaptive state initialization, \( \rho_{\omega}(s_0) \) is represented by a parametric model composed of independent Gaussian distributions over the character state. The Gaussian components are positioned at uniform points along the phase of the motion. To sample from this distribution, we first partition the state features \( s = [\hat{s}, \hat{\phi}] \), where \( \hat{\phi} \) is the phase variable and \( \hat{s} \) represents the other features. The distribution \( \rho_{\omega}(s) \) is then factorized according to:

\[
\rho_{\omega}(s) = \rho_{\omega}(\hat{s}|\hat{\phi})p(\hat{\phi}),
\]

with \( p(\hat{\phi}) \) being a uniform distribution over discrete phase values \([\phi^0, \phi^1, ..., \phi^{d-1}]\). Each phase-conditioned state distribution \( \rho_{\omega}(\hat{s}|\hat{\phi}) \), corresponding to \( \hat{\phi} \), is modeled as a Gaussian \( N(\mu^i, \Sigma^i) \), with mean \( \mu^i \) and diagonal covariance matrix \( \Sigma^i \). The parameters of the initial state distribution consists of the parameters for each Gaussian component \( \omega = \{\mu^i, \Sigma^i\}_{i=0}^{d-1} \). Both the mean and covariance matrix of each component are learned using policy gradients. When sampling an initial state, a phase value is first sampled from the discrete distribution \( p(\hat{\phi}) \). Next, \( \hat{s} \) is sampled from \( \rho_{\omega}(\hat{s}|\hat{\phi}) \), and the combined features constitute the initial state.

10 RESULTS

Motions from the trained policies are best seen in the supplementary video. Figures 4, 6, and 7 compare snapshots of the simulated characters with the original video demonstrations. All video clips were collected from YouTube. The clips depict human actors performing various acrobatic stunts (e.g., flips and cartwheels) and locomotion skills (walking and running). The clips were selected such that the camera is primarily stationary over the course of the motion, and only a single actor is present in the scene. Each clip is trimmed to contain only the relevant portions of their respective motion, and depicts one demonstration of a particular skill.

Table 1 summarizes the performance of the final policies, and a comprehensive set of learning curves are available in the supplementary material. Performance is recorded as the average normalized return over multiple episodes. As it is challenging to directly quantify the difference between the motion of the actor in the video and the simulated character, performance is recorded with respect to the reconstructed reference motion. Since the reference motions recovered from the video clips may not be physically correct, a maximum return of 1 may not be achievable. Nonetheless, given a single video demonstration of each skill, the policies are able to reproduce a large variety of challenging skills ranging from contact-rich motions, such as rolling, to motions with significant flight phases, such as flips and spins. Policies can also be trained to perform skills that require more coordinated interactions with the environment, such as vaulting and pushing a large object.

Retargeting: One of the advantages of physics-based character animation is its ability to synthesize behaviours for novel situations that are not present in the original data. In addition to reproducing the various skills, our framework is also able to retarget the skills to characters and environments that differ substantially from what is presented in the video demonstrations. Since the same simulated character is trained to imitate motions from different human actors, the morpholgy of the character tends to differ drastically from that of the actor. To demonstrated the system’s robustness to morphological discrepancies, we also trained a simulated Atlas robot to imitate a variety of video clips. The proportions of the Atlas’ limbs differ significantly from normal human proportions, and with a weight of 169.5kg, it is considerably heavier than the average human. Despite these drastic differences in morphology, our framework is able to

\begin{table}
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Skill & \( T_{\text{cycle}} \) (s) & \( N_{\text{samples}} \) (10^6) & \( NR \) \\
\hline
Backflip A & 2.13 & 146 & 0.741 \\
Backflip B & 1.87 & 198 & 0.653 \\
Cartwheel A & 2.97 & 136 & 0.824 \\
Cartwheel B & 2.63 & 147 & 0.732 \\
Dance & 2.20 & 257 & 0.631 \\
Frontflip & 1.57 & 126 & 0.708 \\
Gangnam Style & 1.03 & 97 & 0.657 \\
Handspring A & 1.83 & 155 & 0.696 \\
Handspring B & 1.47 & 311 & 0.578 \\
Jump & 2.40 & 167 & 0.653 \\
Jumping Jack & 0.97 & 122 & 0.893 \\
Kick & 1.27 & 158 & 0.761 \\
Kip-Up & 1.87 & 123 & 0.788 \\
Punch & 1.17 & 115 & 0.831 \\
Push & 1.10 & 225 & 0.487 \\
Roll & 2.07 & 122 & 0.603 \\
Run & 0.73 & 126 & 0.878 \\
Spin & 1.07 & 146 & 0.779 \\
Spinkick & 1.87 & 196 & 0.747 \\
Vault & 1.43 & 107 & 0.730 \\
Walk & 0.87 & 122 & 0.952 \\
Atlas: Backflip A & 2.13 & 177 & 0.318 \\
Atlas: Cartwheel A & 2.97 & 174 & 0.456 \\
Atlas: Dance & 2.20 & 141 & 0.324 \\
Atlas: Handspring A & 1.83 & 115 & 0.360 \\
Atlas: Jump & 2.40 & 134 & 0.508 \\
Atlas: Run & 0.73 & 130 & 0.881 \\
Atlas: Vault & 1.43 & 112 & 0.752 \\
Atlas: Walk & 0.87 & 172 & 0.926 \\
\hline
\end{tabular}
\caption{Performance statistics of over 20 skills learned by our framework. \( T_{\text{cycle}} \) denotes the length of the clip. \( N_{\text{samples}} \) records the number of samples collected to train each policy. \( NR \) represents the average normalized return of the final policy, with 0 and 1 being the minimum and maximum possible return per episode respectively. For cyclic skills, the episode horizon is set to 20s. For acyclic skills, the horizon is determined by \( T_{\text{cycle}} \). All statistics are recorded from the humanoid character unless stated otherwise.}
\end{table}
Fig. 6. Simulated characters performing skills learned from video clips. **Top:** Video clip. **Middle:** 3D pose estimator. **Bottom:** Simulated character.
In addition to retargeting to difference morphologies, the skills can also be adapted to different environments. While the video demonstrations were recorded on flat terrain, our framework is able to train policies to perform the skills on irregular terrain. Figure 8 highlights some of the skills that were adapted to environments composed of randomly generated slopes or gaps. The pushing skill can also be retargeted to push a 50 kg box uphill and downhill with a slope of 15%. To enable the policies to perceive their environment, we follow the architecture used by Peng et al. [2018], where a heightmap of the surrounding terrain is included in the input state, and the networks are augmented with corresponding convolutional layers to process the heightmap. Given a single demonstration of an actor performing a backflip on flat terrain, the policy is able to develop strategies for performing a backflip on randomly varying slopes. Similarly, the cartwheel policy learns to carefully coordinate the placement of the hands and feet to avoid falling into the gaps.

In addition to retraining to difference morphologies, the skills can also be adapted to different environments. While the video demonstrations were recorded on flat terrain, our framework is able to train policies to perform the skills on irregular terrain. Figure 8 highlights some of the skills that were adapted to environments composed of randomly generated slopes or gaps. The pushing skill can also be retargeted to push a 50 kg box uphill and downhill with a slope of 15%. To enable the policies to perceive their environment, we follow the architecture used by Peng et al. [2018], where a heightmap of the surrounding terrain is included in the input state, and the networks are augmented with corresponding convolutional layers to process the heightmap. Given a single demonstration of an actor performing a backflip on flat terrain, the policy is able to develop strategies for performing a backflip on randomly varying slopes. Similarly, the cartwheel policy learns to carefully coordinate the placement of the hands and feet to avoid falling into the gaps.

Initial State Distribution: To evaluate the impact of adaptive state initialization (ASI), we compare the performance of policies trained with ASI to those trained with fixed state initialization (FSI) and reference state initialization (RSI). In the case of fixed state initialization, the character is always initialized to the same pose at the start of the motion. With reference state initialization, initial states are sampled randomly from the reference motion as proposed by Peng et al. [2018]. For ASI, the initial state distribution is modeled as a collection of $k=10$ independent Gaussian distributions positioned

Table 2. Performance of policies trained with different initial state distributions. ASI outperforms the other methods for all skills evaluated.

<table>
<thead>
<tr>
<th>Skill</th>
<th>FSI</th>
<th>RSI</th>
<th>ASI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backflip A</td>
<td>0.086</td>
<td>0.602</td>
<td>0.741</td>
</tr>
<tr>
<td>Cartwheel A</td>
<td>0.362</td>
<td>0.738</td>
<td>0.824</td>
</tr>
<tr>
<td>Frontflip</td>
<td>0.435</td>
<td>0.658</td>
<td>0.708</td>
</tr>
<tr>
<td>Handspring A</td>
<td>0.358</td>
<td>0.464</td>
<td>0.696</td>
</tr>
</tbody>
</table>
at uniformly spaced phase values. The mean of each Gaussian is initialized to the state at the corresponding phase of the reference motion, and the diagonal covariance matrix is initialized with the sample covariance of the states from the entire reference motion. Both the mean and covariance matrix of each distribution are then learned through the training process, while the corresponding phase for each distribution is kept fixed. Figure 9 compares the learning curves using the three different methods and Table 2 compares the performance of the final policies. Each result is averaged over three independent runs with different random seeds.

Overall, the behaviour of the learning algorithm appears consistent across multiple runs. Policies trained with ASI consistently outperform their counterparts, converging to the highest return between the different methods. For more challenging skills, such as the backflip and frontflip, ASI also shows notable improvements in learning speed. Policies trained with FSI struggles to reproduce any of the skills. Furthermore, we evaluate the sensitivity of ASI to different numbers of Gaussian components. Policies were trained using $k = 5, 10, 20$ components and their corresponding learning curves are available in Figure 12. Using different numbers of components does not seem to have a significant impact on the performance.
Fig. 12. Learning curves of policies trained with ASI using different number of Gaussian components. The choice of the number of components does not appear to have a significant impact on performance.

Fig. 13. Learning curves comparing policies trained with and without motion reconstruction (MR). MR improves performance for both RSI and ASI.

Table 3. Performance of policies with and without motion reconstruction.

<table>
<thead>
<tr>
<th>Skill</th>
<th>RSI</th>
<th>RSI + MR</th>
<th>ASI</th>
<th>ASI + MR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontflip</td>
<td>0.404</td>
<td>0.658</td>
<td>0.403</td>
<td>0.708</td>
</tr>
<tr>
<td>Handspring A</td>
<td>0.391</td>
<td>0.464</td>
<td>0.631</td>
<td>0.696</td>
</tr>
</tbody>
</table>

of ASI. Qualitatively, the resulting motions also appear similar. The experiments suggest that ASI is fairly robust to different choices for this hyperparameter.

Reference Motion: A policy’s ability to reproduce a video demonstration relies on the quality of the reconstructed reference motion. Here, we investigate the effects of rotation augmentation and motion reconstruction on the resulting reference motions. Most existing datasets of human poses are biased heavily towards upright poses. However, an actor’s orientation can vary more drastically when performing highly dynamic and acrobatic skills, e.g. upside-down poses during a flip. Rotation augmentation significantly improves predictions for these less common poses. Figure 10 compares the predictions from pose estimators trained with and without rotation augmentation. We found that this step is vital for accurate predictions of more extreme poses, such as those present in the backflip and handspring. Without augmentation, both pose estimators consistently fail to predict upside-down poses.

Next, we evaluate the effects of the motion reconstruction stage in producing reference motions that can be better reproduced by a simulated character. Policies that are trained to imitate the optimized reference motions generated by motion reconstruction (MR), are compared to policies trained without MR, where the poses from the 3D pose estimator are directly used as the reference motion. We found that this step is vital for accurate predictions of more extreme poses, such as those present in the backflip and handspring. Without augmentation, both pose estimators consistently fail to predict upside-down poses.

Learning curves comparing policies trained using reference motions before and after motion reconstruction are available in Figure 13, and Table 3 summarizes the performance of the final policies. For each type of reference motion, we also compared policies trained with either RSI or ASI. Overall, imitating reference motions generated by the motion reconstruction processes improves performance and learning speed for the different skills. The improvements due to MR appears more pronounced when policies are trained with RSI. Since the initial states are sampled directly from the reference motion, performance is more susceptible to artifacts present in the reference motion. MR also shows consistent improvement across multiple training runs when using ASI. Note that since the reward reflects similarly to the reference motion, and not the original video, a higher return does not necessarily imply better reproduction of the video demonstration. Instead, the higher return with MR indicates that the simulated character is able to better reproduce the reference motions produced by MR than the raw predictions from the pose estimator. Thus, the results suggest that by enforcing temporal consistency and mitigating artifacts due to inaccurate pose predictions, motion reconstruction is able to generate reference motions that are more amenable to being mimicked by a simulated character.

Motion Completion: The motion completion technique is demonstrated on images depicting actors performing various acrobatic skills, such as backflips, cartwheels, and rolls. Figure 14 illustrates some of the predicted motions. Only a single image is provided during evaluation, and the system is then able to predict a plausible future motion for the depicted actor. The robustness of the learned policies enable the character to synthesize plausible behaviours even when the actor’s pose \( \bar{q} \) differs significantly from the reference motions used to train the policies. However, the system can fail to generate reasonable motions for an image if the actor’s pose is drastically different from those of the reference motions, and the predictions are limited to skills spanned by the existing policies.
We presented a framework for learning full-body motion skills from monocular video demonstrations. Our method is able to reproduce a diverse set of highly dynamic and acrobatic skills with simulated humanoid characters. We proposed a data augmentation technique that improves the performance of pose estimators for challenging acrobatic motions, and a motion reconstruction method that leverages an ensemble of pose estimators to produce higher-fidelity reference motions. Our adaptive state initialization method substantially improves the performance of the motion imitation process when imitating low-fidelity reference motions. Our framework is also able to re-target skills to characters and environments that differ drastically from those present in the original video clips. By leveraging a library of learned controllers, we introduced a physics-based motion completion technique, where given a single image of a human actor, our system is able to predict plausible future motions of the actor.

While our framework is able to imitate a diverse collection of video clips, it does have a number of limitations. Since the success of the motion imitation stage depends on the accuracy of the reconstructed motion, when the pose estimators are not able to correctly predict an actor’s pose, the resulting policy will fail to reproduce the behavior. Examples include the kid-up, where the reconstructed motions did not accurately capture the motion of the actor’s arms, and the spin kick, where the pose estimator did not capture the extension of the actor’s leg during the kick. Furthermore, our characters still sometimes exhibit artifacts such as peculiar postures and stiff movements. Fast dance steps, such as those exhibited in the Gangnam Style clip, remains challenging for the system, and we have yet to be able to train policies that can closely reproduce such nimble motions. Due to difficulties in estimating the global translation of the character’s root, our results have primarily been limited to video clips with minimal camera motion.

Nonetheless, we believe this work opens many exciting directions for future exploration. Our experiments suggest that learning highly-dynamic skills from video demonstrations is achievable by building on state-of-the-art techniques from computer vision and reinforcement learning. An advantage of our modular design is that new advances relevant to the various stages of the pipeline can be readily incorporated to improve the overall effectiveness of the framework. However, an exciting direction for future work is to investigate methods for more end-to-end learning from visual demonstrations, for example taking inspiration from Sermanet et al. [2017] and Yu et al. [2018a], which may reduce the dependence on accurate pose estimators. Another exciting direction is to capitalize on our method’s ability to learn from video clips and focus on large, outdoor activities, as well as motions of nonhuman animals that are conventionally very difficult, if not impossible, to mocap.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful feedback, and AWS for providing computational resources. This research was funded by an NSERC Postgraduate Scholarship, a Berkeley Fellowship for Graduate Study, and BAIR.

REFERENCES


Xingyi Zhou, Qixing Huang, Xiao Sun, Xianguang Xue, and Yichen Wei. 2017. Weakly-supervised transfer for 3D Human Pose Estimation in the Wild. In *IEEE International Conference on Computer Vision, ICCV.*


SUPPLEMENTARY MATERIAL

Table 4. Summary of Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{x}_t$</td>
<td>2D joint prediction at frame $t$</td>
</tr>
<tr>
<td>$\hat{q}_t$</td>
<td>3D pose prediction at frame $t$</td>
</tr>
<tr>
<td>$z_t$</td>
<td>HMR embedding of the image frame at $t$</td>
</tr>
<tr>
<td>$q(z_t)$</td>
<td>3D pose as a function of the embedding</td>
</tr>
<tr>
<td>$F_j(\cdot)$</td>
<td>Forward kinematics function that computes the 3D position of joint $j$</td>
</tr>
<tr>
<td>$q^t$</td>
<td>Final 3D reference pose after motion reconstruction</td>
</tr>
<tr>
<td>$s_t$</td>
<td>state of the simulated character at timestep $t$</td>
</tr>
<tr>
<td>$a_t$</td>
<td>action</td>
</tr>
<tr>
<td>$r_t$</td>
<td>reward</td>
</tr>
<tr>
<td>$R_t$</td>
<td>return starting at timestep $t$, $\sum_{t=0}^{T-1} y^t r_{t+1}$</td>
</tr>
<tr>
<td>$\mathcal{A}_t$</td>
<td>advantage at timestep $t$, $R_t - V(s_t)$</td>
</tr>
<tr>
<td>$\tau$</td>
<td>a simulated trajectory $(s_0, a_0, r_0, s_1, \ldots, s_{T-1}, s_T)$</td>
</tr>
<tr>
<td>$\pi_q(a</td>
<td>s)$</td>
</tr>
<tr>
<td>$\rho_{\omega}(s_0)$</td>
<td>initial state distribution with parameters $\omega$</td>
</tr>
<tr>
<td>$\alpha_V$</td>
<td>value stepsize</td>
</tr>
<tr>
<td>$\omega$</td>
<td>initial state distribution stepsize</td>
</tr>
</tbody>
</table>

A LEARNING ALGORITHM

When training with adaptive state initialization, the multi-agent objective is given by:

$$ J(\theta, \omega) = \mathbb{E}_{\tau \sim p_{\omega,s_0}(\tau)} \left[ \sum_{t=0}^{T} y^t r_t \right] $$

$$ = \int_{\tau} \left( \rho_{\omega}(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t) \pi_q(a_t|s_t) \right) \left( \sum_{t=0}^{T} y^t r_t \right) d\tau. $$

The policy gradient of the initial state distribution $\rho_{\omega}(s_0)$ can then be as follows:

$$ \nabla_\omega J(\theta, \omega) = \int_{\tau} \left( \nabla_\omega \rho_{\omega}(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t) \pi_q(a_t|s_t) \right) \left( \sum_{t=0}^{T} y^t r_t \right) d\tau $$

$$ = \int_{\tau} \left( \rho_{\omega}(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t) \pi_q(a_t|s_t) \right) \frac{\partial}{\partial \rho_{\omega}(s_0)} \left( \sum_{t=0}^{T} y^t r_t \right) d\tau $$

$$ = \int_{\tau} \left( \nabla_\omega \log \left( \rho_{\omega}(s_0) \right) \rho_{\omega}(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t) \pi_q(a_t|s_t) \right) \left( \sum_{t=0}^{T} y^t r_t \right) d\tau $$

$$ = \mathbb{E}_{\tau \sim p_{\omega,s_0}(\tau)} \left[ \nabla_\omega \log \left( \rho_{\omega}(s_0) \right) \sum_{t=0}^{T} y^t r_t \right]. $$

Algorithm 1 provides an overview of the training process. Policy updates are performed after a batch of 4096 samples has been collected, and minibatches of size 256 are then sampled from the data for each gradient step. The initial state distribution is updated using batches of 2000 episodes. Only one gradient step is calculated per batch when updating the initial state distribution. SGD with a stepsize of $\alpha_V = 2.5 \times 10^{-6}$ and momentum of 0.9 is used for the policy network, a stepsize of $\alpha_R = 0.01$ is used for the value network, and a stepsize of $\alpha_\rho = 0.001$ is used for the initial state distribution. The discount factor $\gamma$ is set to 0.95, and $\lambda$ is set to 0.95 for TD($\lambda$) and GAE($\lambda$). PPO with a clipping threshold of 0.2 is used for policy updates.
Fig. 15. Learning curves of policies trained for the humanoid. Performance is recorded as the average normalized return over multiple episodes, with 0 representing the minimum possible return and 1 being the maximum.
Fig. 16. Learning curves of policies trained for the Atlas.