SFV: Reinforcement Learning of Physical Skills from Videos

XUE BIN PENG, University of California, Berkeley
ANGJOO KANAZAWA, University of California, Berkeley
JITENDRA MALIK, University of California, Berkeley
PIETER ABBEEL, University of California, Berkeley
SERGEY LEVINE, University of California, Berkeley

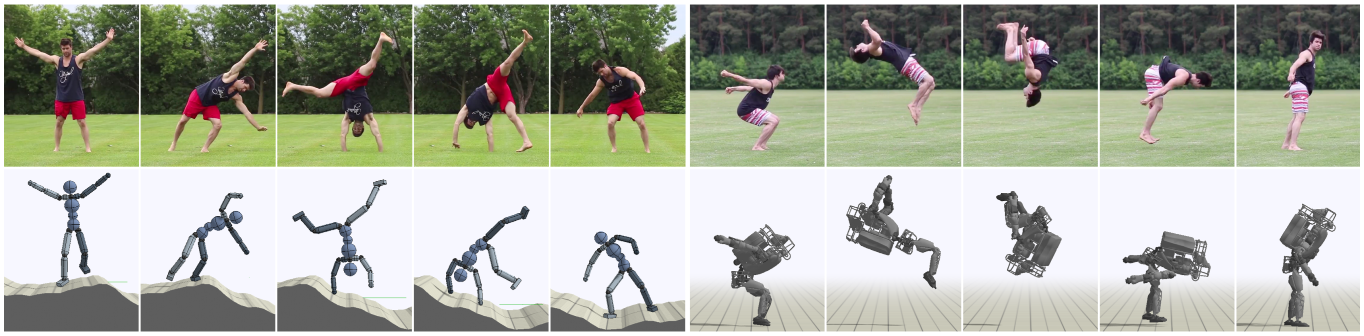


Fig. 1. Simulated characters performing highly dynamic skills learned by imitating video clips of human demonstrations. **Left:** Humanoid performing cartwheel B on irregular terrain. **Right:** Backflip A retargeted to a simulated Atlas robot.

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. (Video[1])

CCS Concepts: • **Computing methodologies** → **Animation**; *Physical simulation*; *Control methods*; *Reinforcement learning*; *Tracking*;

[1]https://xbpeng.github.io/projects/SFV/index.html

Authors' addresses: Xue Bin Peng, University of California, Berkeley; Angjoo Kanazawa, University of California, Berkeley; Jitendra Malik, University of California, Berkeley; Pieter Abbeel, University of California, Berkeley; Sergey Levine, University of California, Berkeley.

Additional Key Words and Phrases: physics-based character animation, computer vision, video imitation, reinforcement learning, motion reconstruction

ACM Reference Format:
Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. 2018. SFV: Reinforcement Learning of Physical Skills from Videos. *ACM Trans. Graph.* 37, 6, Article 178 (November 2018), 17 pages. https://doi.org/10.1145/3272127.3275014

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 *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 lower-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. 2015b]. 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 behaviours. 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 behaviours. 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 quadruped 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 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 $\{q_t^*\}$ is optimized such that

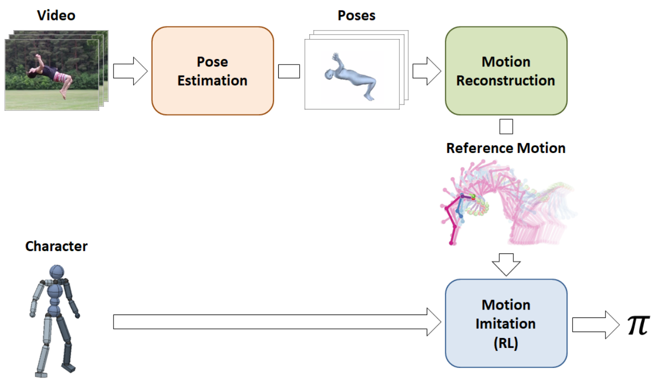


Fig. 2. The pipeline consists of three stages: pose estimation, motion reconstruction, and imitation. It receives as input, a video clip of an actor performing a particular skill and a simulated character model, and outputs a control policy that enables the character to reproduce the skill in simulation.

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 π 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 Π. 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; Peng 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 $a \in A$ 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 π. The environment then responds with a successor state $s' = s_{t+1}$, sampled from the dynamics $p(s'|s,a)$, and a scalar reward r_t, which reflects the desirability of the transition. For a parametric policy $\pi_\theta(a|s)$, with parameters θ, the goal of the agent is to learn the optimal parameters θ^* that maximizes its expected return

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

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 $\tau = (s_0, a_0, s_1, ..., a_{T-1}, s_T)$ induced by the policy π_θ, 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 $\triangledown_\theta J(\theta)$ is estimated using trajectories that are obtained by rolling out the policy:

$$\triangledown_\theta J(\theta) = \mathbb{E}_{s_t \sim d_\theta(s_t), a_t \sim \pi_\theta(a_t|s_t)} \left[\triangledown_\theta \log(\pi_\theta(a_t|s_t)) \mathcal{A}_t \right],$$

where $d_\theta(s_t)$ is the state distribution under the policy π_θ. $\mathcal{A}_t = R_t - V(s_t)$ represents the advantage of taking an action a_t at a given state s_t, with $R_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}$ 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 humans in upright orientations. Thus, off-the-shelf pose estimators struggle to predict the poses in these videos. To compensate for this discrepancy, we augment the standard datasets with rotated

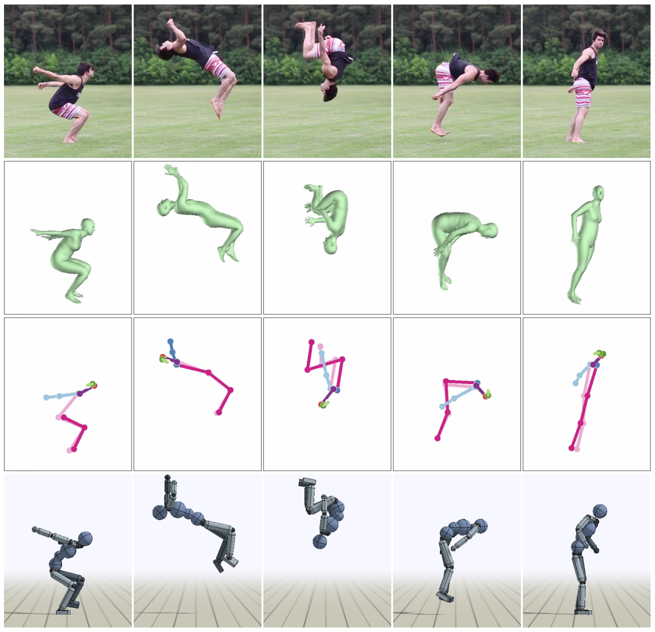


Fig. 3. Comparison of the motions generated by different stages of the pipeline for backflip A. **Top-to-Bottom:** Input video clip, 3D pose estimator, 2D pose estimator, simulated character.

versions of the existing images, where the rotations are sampled uniformly between $[0, 2\pi]$. We found that training the pose estimators with this augmented dataset, substantially improves performance for acrobatic poses. Once trained, both estimators are applied independently to every frame to extract a 2D pose trajectory $\{\hat{x}_t\}$ and 3D pose trajectory $\{\hat{q}_t\}$. Note that the 2D pose $\hat{x}_t$ consists only of the 2D screen coordinates of the actor's joints, but tends to be more accurate than the 3D predictions. Examples of the predictions from the different pose estimators are shown in Figure 3. The independent predictions from the pose estimators are then consolidated in the motion reconstruction stage to produce the final reference motion.

6 MOTION RECONSTRUCTION

Since poses are predicted independently for every frame in the pose estimation stage, simply sequencing the poses into a trajectory tends to produce motions that exhibit artifacts due to inconsistent predictions across adjacent frames (see supplementary video). The motion artifacts often manifest as nonphysical behaviours in the reference motion, such as high-frequency jitter and sudden changes in pose. These artifacts can hinder the simulated character's ability to reproduce the intended motion. The role of the motion reconstruction stage is to take advantage of the predictions from the two pose estimators to reconstruct a new kinematic trajectory that reconciles the individual predictions and mitigates artifacts, such that the resulting reference motion is more amenable for imitation.

Specifically, given the predictions from the 2D and 3D pose estimators, we optimize a 3D pose trajectory that consolidates their predictions while also enforcing temporal consistency between adjacent frames. Instead of directly optimizing in the 3D pose space, we take advantage of the encoder-decoder structure of the 3D pose estimator and optimize the 3D pose trajectory in the latent pose space z_t, which captures the manifold of 3D human poses [Kanazawa et al. 2018]. The final 3D reference motion is constructed by optimizing a trajectory $Z = \{z_t\}$ in the latent space to minimize the reconstruction loss l_{rec}:

$$l_{rec}(Z) = w_{2D} l_{2D}(Z) + w_{3D} l_{3D}(Z) + w_{sm} l_{sm}(Z)$$

$$w_{2D} = 10, w_{3D} = 100, w_{sm} = 25,$$

The 2D consistency loss l_{2D} minimizes the reprojection error between the predicted 2D joint locations and the 2D projections of the corresponding joints arising from the pose specified by z_t

$$l_{2D} = \sum_t \sum_j c_{t,j} \left\| \left(\hat{x}_{t,j} - \Pi \left[F_j \left(q \left(z_t \right) \right) \right] \right) \right\|_1 ,$$

where $\hat{x}_{t,j}$ is the predicted 2D location of the jth joint, $c_{t,j}$ is the confidence of the prediction, and $F_j[\cdot]$ is the forward kinematics function that computes the 3D position of joint j given the 3D pose. $q(z_t)$ represents the pose decoded from z_t, and $\Pi[\cdot]$ is the weak-perspective projection that transforms 3D positions to 2D screen coordinates.

The 3D consistency loss l_{3D} encourages the optimized trajectory to stay close to the initial 3D prediction $\hat{q}_t$:

$$l_{3D} = \sum_t w_t dist(\hat{q}_t, q(z_t)),$$

where $dist(\cdot,\cdot)$ 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 $\delta_t = \sum_j c_{t,j} ||(\hat{x}_{t,j} - \Pi F_j(\hat{q}_t))||_2$. 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 π 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 $\pi_\theta(a_t|s_t)$, 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:

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

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

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:

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

A detailed derivation is available in the supplementary material. Similar to the standard policy gradient for π, 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:

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

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

The pose reward r_t^p 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 $q_{t,j}^*$ represent the rotation of the jth joint from the simulated character and reference motion respectively, $q_1 \ominus q_2$ denotes the quaternion difference, and $||q||$ computes the scalar rotation of a quaternion about its axis:

$$r_t^p = \exp\left[-2 \left(\sum_j ||q_{t,j}^* \ominus q_{t,j}||^2 \right) \right].$$

Similarly, the velocity reward r_t^v is calculated from the difference of local joint velocities, with $\dot{q}_{t,j}$ being the angular velocity of the jth joint. The target velocity $\dot{q}_{t,j}^*$ is computed from the reference motion via finite difference.

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

The end-effector reward r_t^e 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 $e \in$ [left foot, right foot, left hand, right hand]:

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

Finally, r_t^c penalizes deviations in the character's center-of-mass c_t from that of the reference motion c_t^*:

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

The positional qualities $p_{t,e}^*$ and c_t^* are computed from the reference pose q_t^* via forward kinematics.

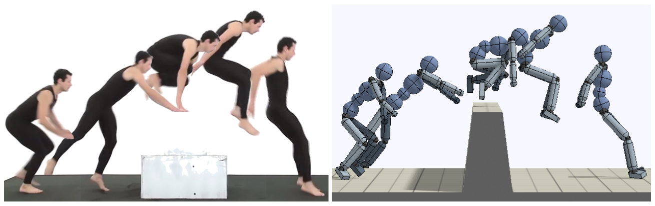


Fig. 4. Character imitating a 2-handed vault.

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 $\rho_\omega(s_0)$. A rollout is then simulated by sampling actions from the policy $\pi_\theta(a_t|s_t)$ 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(λ) is used to compute target values for the value network $V_\psi(s)$ [Sutton and Barto 1998] and GAE(λ) 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 $\{(\{q_t^i\}, \pi^i)\}$, where π^i is the policy trained to imitate reference motion $\{q_t^i\}$. To predict the actor's motion, the target image is first processed by the 3D pose estimator to extract the pose of the actor $\bar{q}$. The extracted pose is then compared to every frame of each reference motion to select the reference motion and corresponding frame $q_{t^*}^{i^*}$ that is most similar to $\bar{q}$,

$$q_{t^*}^{i^*} = \arg\min_{q_t^i} ||\bar{q} \ominus q_t^i||.$$

Next, the simulated character is initialized to the state defined by the pose $\bar{q}$ and, since still images do not provide any velocity information, we initialize the character's velocity to that of the selected reference motion $\dot{q}_{t^*}^{i^*}$. The motion is then simulated by applying the policy π^{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.

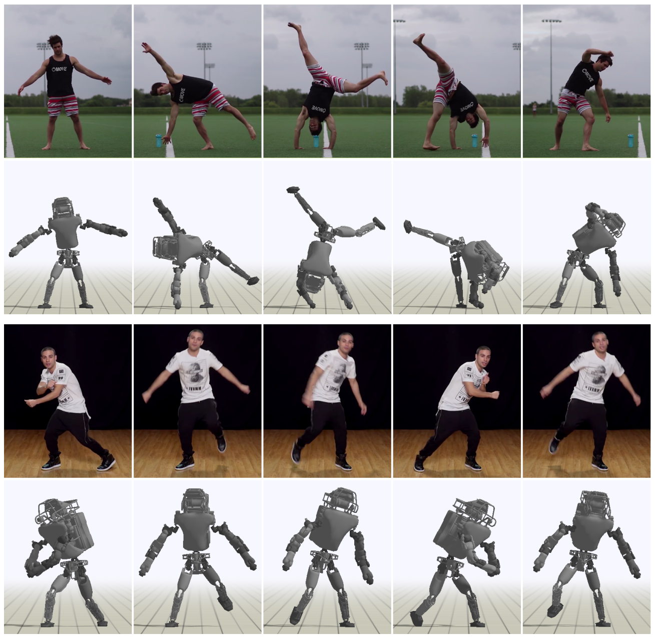


Fig. 5. Simulated Atlas robot performing skills learned from video demonstrations. **Top:** Cartwheel A. **Bottom:** Dance.

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 Σ:

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

The value function $V_\psi(s)$ 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. 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 a 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 s_0 sampled from the initial state distribution $\rho_\omega(s_0)$. When applying adaptive state initialization, $\rho_\omega(s_0)$ is represented with 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 $s = [\hat{s}, \phi]$, where ϕ is the phase variable and $\hat{s}$ represents the other features. The distribution $\rho_\omega(s)$ is then factorized according to:

$$\rho_\omega(s) = p_\omega(\hat{s}|\phi)p(\phi),$$

with $p(\phi)$ being a uniform distribution over discrete phase values $[\phi^0, \phi^1, ..., \phi^{k-1}]$. Each phase-conditioned state distribution $p_\omega(\hat{s}|\phi^i)$, corresponding to ϕ^i, is modeled as a Gaussian $\mathcal{N}(\mu^i, \Sigma^i)$, with mean μ^i and diagonal covariance matrix Σ^i. The parameters of the initial state distribution consists of the parameters for each Gaussian component $\omega = \{\mu^i, \Sigma^i\}_{i=0}^{k-1}$. 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 $p(\phi)$. Next, $\hat{s}$ is sampled from $p_\omega(\hat{s}|\phi)$, and the combined features constitute the initial state.

10 RESULTS

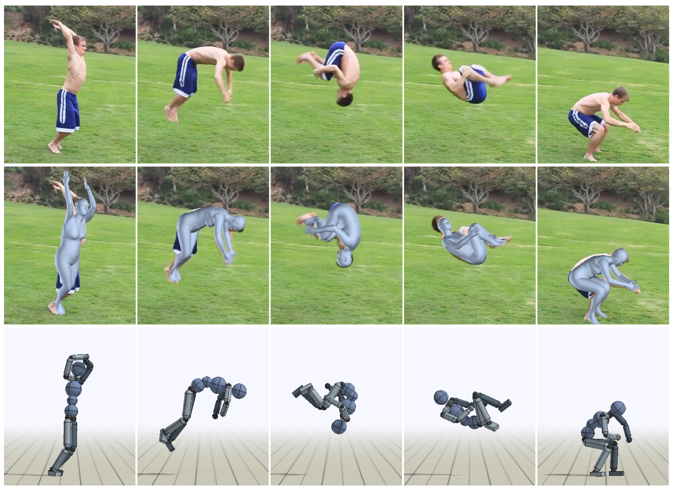
Motions from the trained policies are best seen in the supplementary video. Figures 4, 6, and 7 compare snapshots of the simulated characters with the original video demonstrations. All video clips were collected from YouTube. The clips depict human actors performing various acrobatic stunts (e.g. flips 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.

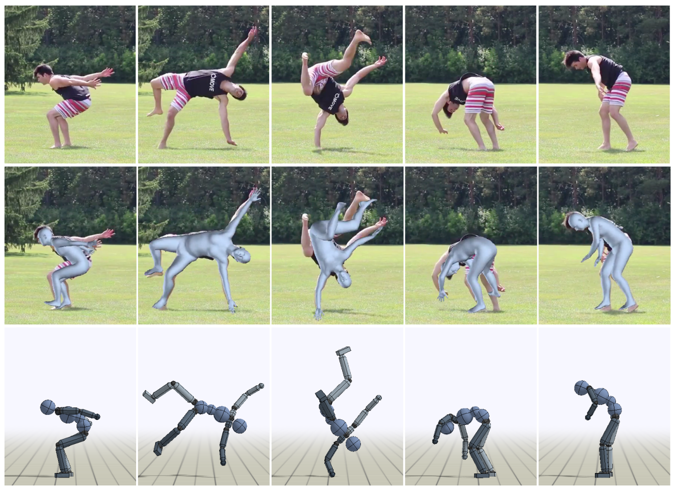
Table 1 summarizes the performance of the final policies, and a comprehensive set of learning curves are available in the supplementary material. Performance is recorded as the average normalized return over multiple episodes. As it is challenging to directly quantify the difference between the motion 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.

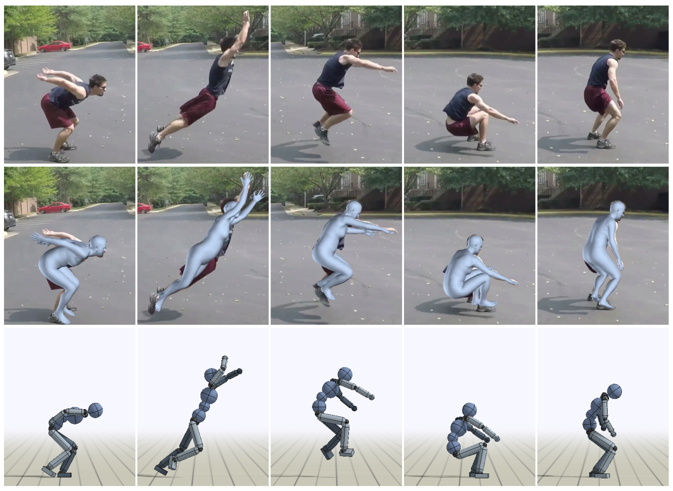
Table 1. Performance statistics of over 20 skills learned by our framework. T_{cycle} denotes the length of the clip. $N_{samples}$ records the number of samples collected to train each policy. NR represents the average normalized return of the final policy, with 0 and 1 being the minimum and maximum possible return per episode respectively. For cyclic skills, the episode horizon is set to $20s$. For acyclic skills, the horizon is determined by T_{cycle}. All statistics are recorded from the humanoid character unless stated otherwise.

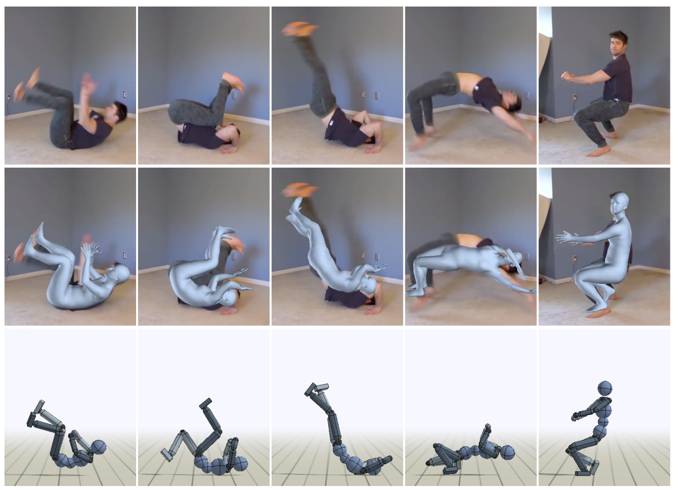
Skill	T_{cycle} **(s)**	$N_{samples}$ (10^6)	NR
Backflip A	2.13	146	0.741
Backflip B	1.87	198	0.653
Cartwheel A	2.97	136	0.824
Cartwheel B	2.63	147	0.732
Dance	2.20	257	0.631
Frontflip	1.57	126	0.708
Gangnam Style	1.03	97	0.657
Handspring A	1.83	155	0.696
Handspring B	1.47	311	0.578
Jump	2.40	167	0.653
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	0.932
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

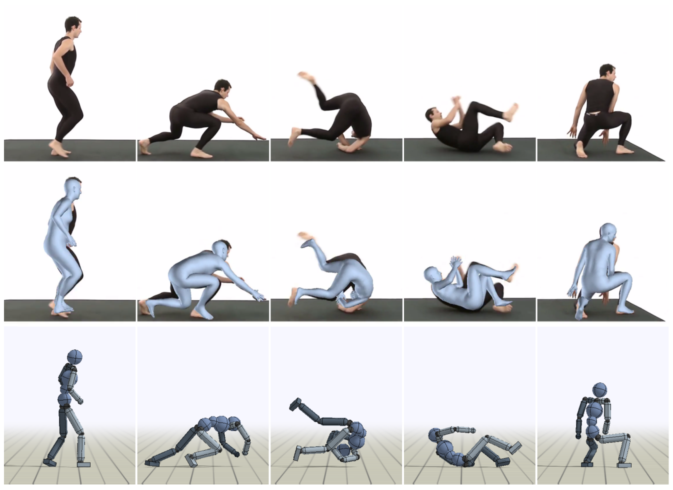
Retargeting: One of the advantages of physics-based character animation is its ability to synthesize behaviours for novel situations that are not present in the original data. In addition to reproducing the various skills, our framework is also able to retarget the skills to characters and environments 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 morphology 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


(a) Frontflip


(b) Handspring A


(c) Jump


(d) Kip-Up


(e) Roll

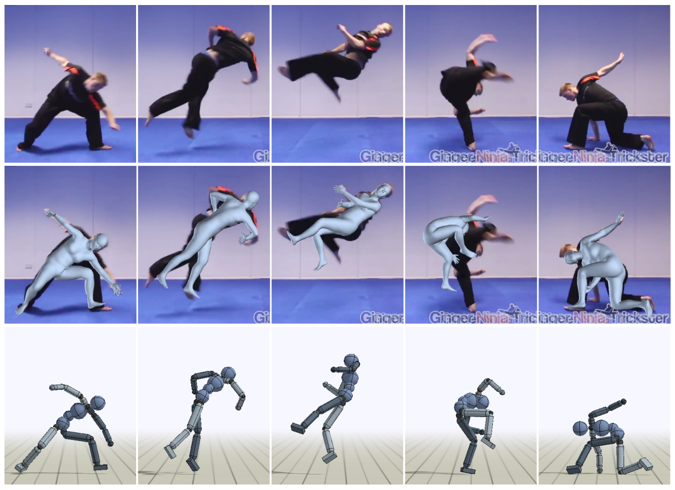

(f) Spin

Fig. 6. Simulated characters performing skills learned from video clips. **Top:** Video clip. **Middle:** 3D pose estimator. **Bottom:** Simulated character.

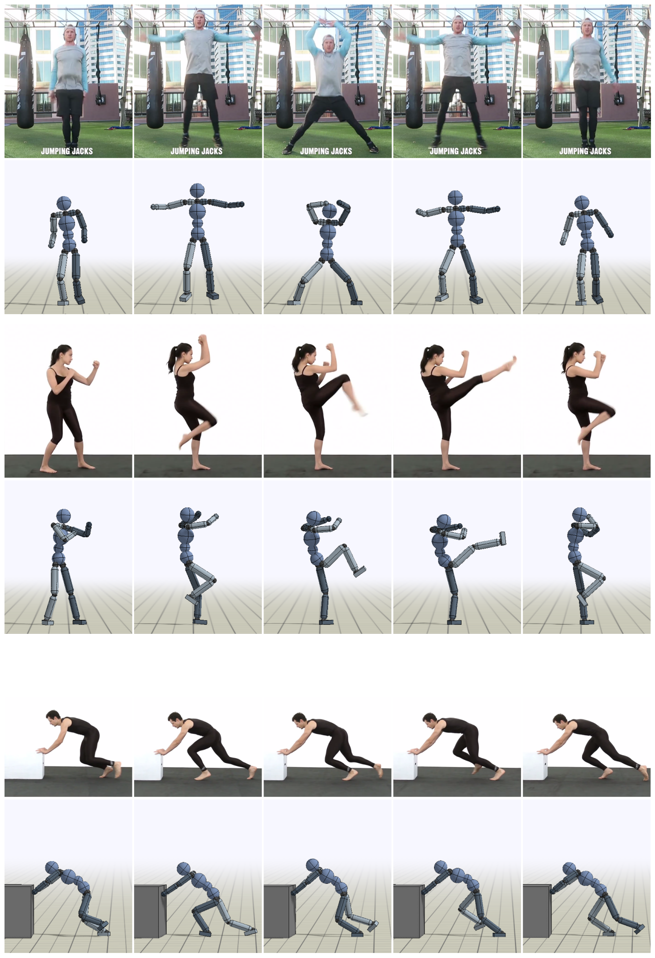


Fig. 7. Humanoid character imitating skills from video demonstrations. **Top-to-Bottom:** Jumping jack, kick, push.

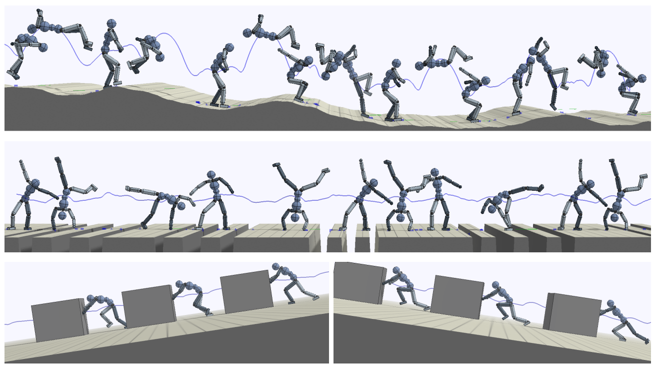


Fig. 8. Skills retargeted to different environments. **Top-to-Bottom:** Backflip A across slopes, cartwheel B across gaps, pushing a box downhill and uphill.

learn policies that enable the Atlas to reproduce a diverse set of challenging skills. Table 1 summarizes the performance of the Atlas policies, and Figure 5 illustrates snapshots of the simulated motions.

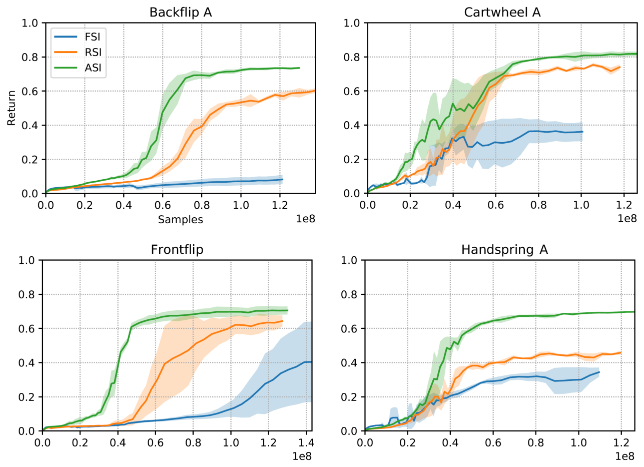


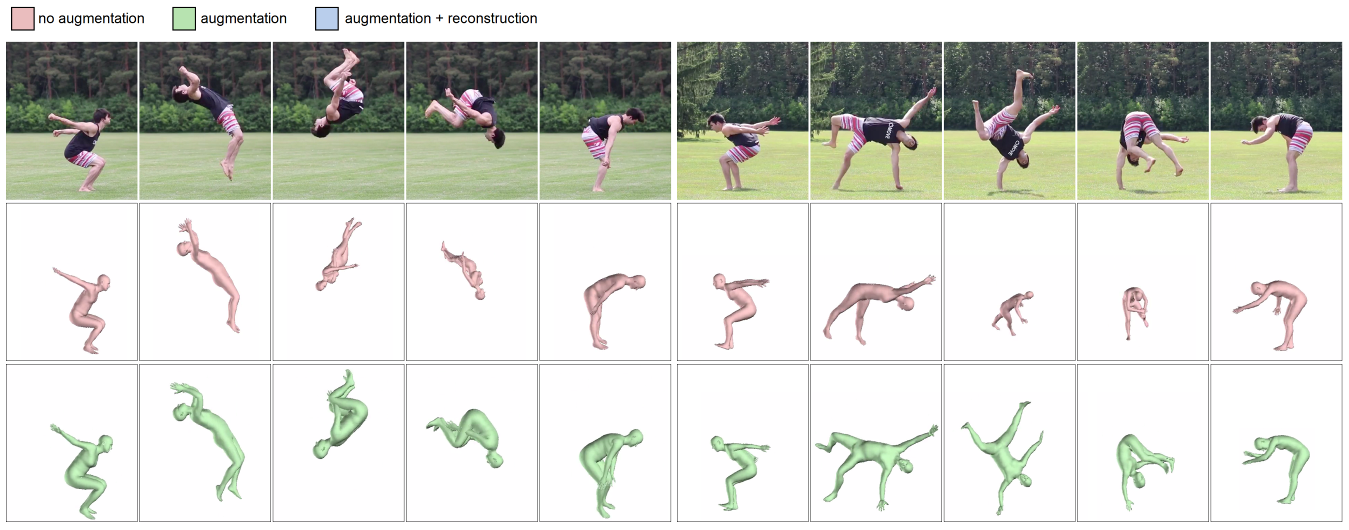
Fig. 9. Learning curves comparing policies trained with fixed state initialization (FSI), reference state initialization (RSI), and adaptive state initialization (ASI). Three policies initialized with different random seeds are trained for each combination of skill and initial state distribution. Compared to its counterparts, ASI consistently improves performance and learning speed.

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

Skill	**FSI**	**RSI**	**ASI**
Backflip A	0.086	0.602	**0.741**
Cartwheel A	0.362	0.738	**0.824**
Frontflip	0.435	0.658	**0.708**
Handspring A	0.358	0.464	**0.696**

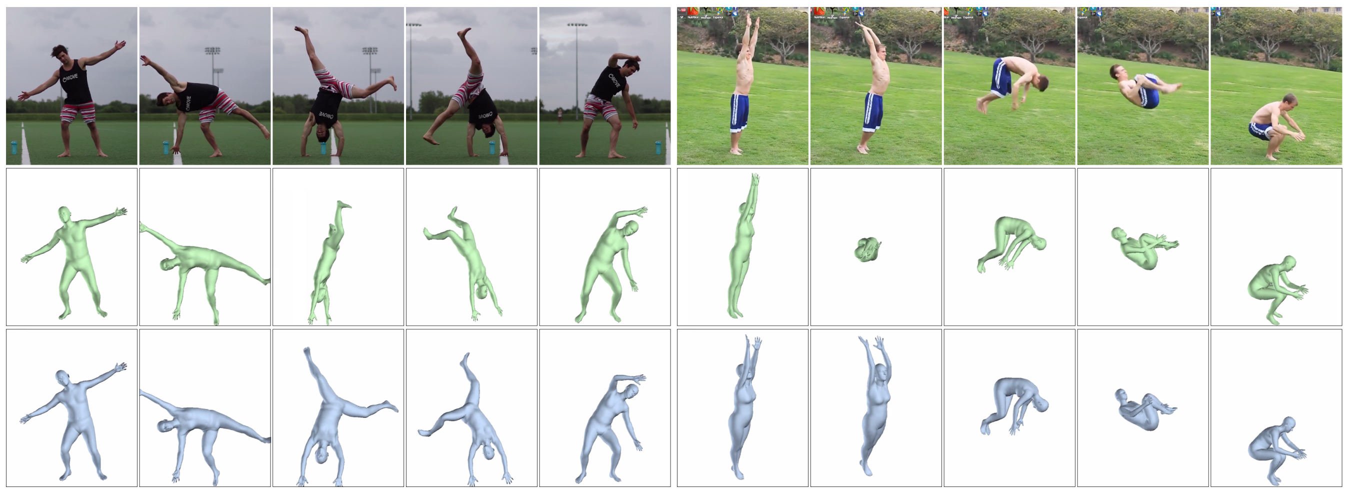
In addition to retargeting to difference morphologies, the skills can also be adapted to different environments. While the video demonstrations were recorded on flat terrain, our framework is able to train policies to perform the skills on irregular terrain. Figure 8 highlights some of the skills that were adapted to environments composed of randomly generated slopes or gaps. The pushing skill can also be retargeted to push a $50kg$ 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



(a) Backflip A (b) Handspring A

Fig. 10. 3D pose predictions with and without rotation augmentation. **Top:** Video. **Middle:** Without rotation augmentation. **Bottom:** With rotation augmentation. The pose estimator trained without rotation augmentation fails to correctly predict challenging poses, such as when the actor is upside-down.



(a) Cartwheel A (b) Frontflip

Fig. 11. 3D pose predictions before and after motion reconstruction. **Top:** Video. **Middle:** Raw predictions from the 3D pose estimator before motion reconstruction. **Bottom:** After motion reconstruction. The motion reconstruction process is able to fix erroneous predictions from the 3D pose estimator by taking advantage of the information from the 2D pose estimator and temporal consistency between adjacent frames.

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

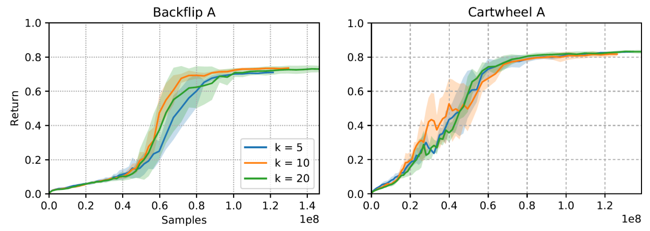


Fig. 12. Learning curves of policies trained with ASI using different number of Gaussian components. The choice of the number of components does not appear to have a significant impact on performance.

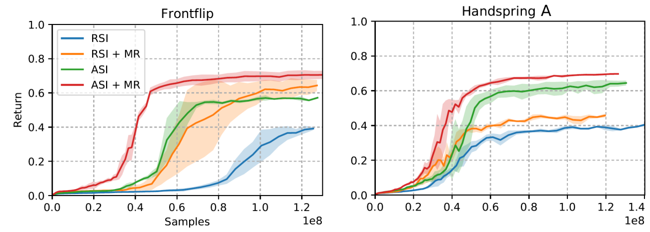


Fig. 13. Learning curves comparing policies trained with and without motion reconstruction (MR). MR improves performance for both RSI and ASI.

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

Skill	RSI	RSI + MR	ASI	ASI + MR
Frontflip	0.404	0.658	0.403	**0.708**
Handspring A	0.391	0.464	0.631	**0.696**

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. Polices 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. Figure 11 compares the motions before and after MR. While the 3D pose estimator occasionally produces erroneous predictions, the MR process is able to correct these errors by taking advantage of the predictions from the 2D estimator and enforcing temporal consistency between adjacent frames. 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.

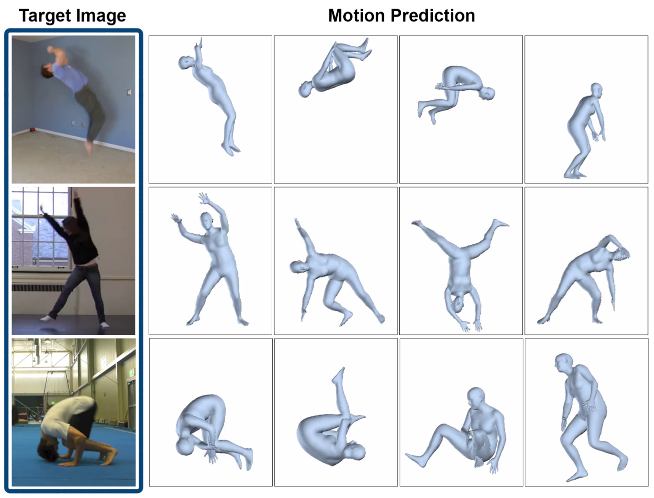


Fig. 14. Given a target image, our motion completion technique predicts plausible motions for the actor in the image.

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.

11 DISCUSSION AND LIMITATIONS

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 retarget 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 kip-up, where the reconstructed motions did not accurately capture the motion of the actor's arms, and the spinkick, 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

Salman Aslam. 2018. YouTube by the Numbers. https://www.omnicoreagency.com/youtube-statistics/. (2018). Accessed: 2018-05-15.

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. 2016. Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. In *European Conference on Computer Vision, ECCV (Lecture Notes in Computer Science)*. Springer International Publishing.

Christoph Bregler and Jitendra Malik. 1998. Tracking people with twists and exponential maps. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE, 8–15.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *CoRR* abs/1606.01540 (2016). arXiv:1606.01540

Bullet. 2015. Bullet Physics Library. (2015). http://bulletphysics.org.

Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *CVPR*.

CMU. 2018. CMU Graphics Lab Motion Capture Database. (2018). http://mocap.cs.cmu.edu.

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2009. Robust Task-based Control Policies for Physics-based Characters. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 28, 5 (2009), Article 170.

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized Biped Walking Control. *ACM Transctions on Graphics* 29, 4 (2010), Article 130.

Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel van de Panne. 2011. Locomotion Skills for Simulated Quadrupeds. *ACM Transactions on Graphics* 30, 4 (2011), Article TBD.

Marco da Silva, Yeuhi Abe, and Jovan Popović. 2008. Interactive Simulation of Stylized Human Locomotion. In *ACM SIGGRAPH 2008 Papers (SIGGRAPH '08)*. ACM, New York, NY, USA, Article 82, 10 pages. https://doi.org/10.1145/1399504.1360681

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. 2016. Benchmarking Deep Reinforcement Learning for Continuous Control. *CoRR* abs/1604.06778 (2016). arXiv:1604.06778

Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. 2013. Flexible Muscle-Based Locomotion for Bipedal Creatures. *ACM Transactions on Graphics* 32, 6 (2013).

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned Neural Networks for Character Control. *ACM Trans. Graph.* 36, 4, Article 42 (July 2017), 13 pages.

Daniel Holden, Jun Saito, and Taku Komura. 2016. A Deep Learning Framework for Character Motion Synthesis and Editing. *ACM Trans. Graph.* 35, 4, Article 138 (July 2016), 11 pages.

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. 2014. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence, TPAMI* 36, 7 (2014), 1325–1339.

Ronald J. Williams and Jing Peng. 1991. Function Optimization Using Connectionist Reinforcement Learning Algorithms. 3 (09 1991), 241–.

Sam Johnson and Mark Everingham. 2010. Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation. In *Proceedings of the British Machine Vision Conference, BMVC*. 12.1–12.11.

Sham Kakade. 2001. A Natural Policy Gradient. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS'01)*. MIT Press, Cambridge, MA, USA, 1531–1538. http://dl.acm.org/citation.cfm?id=2980539.2980738

Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. 2018. End-to-end Recovery of Human Shape and Pose. In *Computer Vision and Pattern Regognition (CVPR)*.

H. Lee and Z. Chen. 1985. Determination of 3D human body postures from a single view. *Computer Vision Graphics and Image Processing* 30, 2 (1985), 148âĂŞ–168.

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010a. Data-driven Biped Control. In *ACM SIGGRAPH 2010 Papers (SIGGRAPH '10)*. ACM, New York, NY, USA, Article 129, 8 pages.

Yoonsang Lee, Moon Seok Park, Taesoo Kwon, and Jehee Lee. 2014. Locomotion Control for Many-muscle Humanoids. *ACM Trans. Graph.* 33, 6, Article 218 (Nov. 2014), 11 pages.

Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. 2010b. Motion Fields for Interactive Character Locomotion. In *ACM SIGGRAPH Asia 2010 Papers (SIGGRAPH ASIA '10)*. ACM, New York, NY, USA, Article 138, 8 pages.

Sergey Levine, Jack M. Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. 2012. Continuous Character Control with Low-Dimensional Embeddings. *ACM Transactions on Graphics* 31, 4 (2012), 28.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr DollÃąr, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)* (2014-01-01). ZÃijrich. /se3/wp-content/uploads/2014/09/coco_eccv.pdf,http://mscoco.org

Libin Liu and Jessica Hodgins. 2017. Learning to Schedule Control Fragments for Physics-Based Characters Using Deep Q-Learning. *ACM Trans. Graph.* 36, 3, Article 29 (June 2017), 14 pages.

Libin Liu, Michiel van de Panne, and KangKang Yin. 2016. Guided Learning of Control Graphs for Physics-Based Characters. *ACM Transactions on Graphics* 35, 3 (2016).

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based Contact-rich Motion Control. *ACM Transctions on Graphics* 29, 4 (2010), Article 128.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34, 6 (Oct. 2015), 248:1–248:16.

Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. 2017. VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH* 36 (July 2017), 14. http://gvv.mpi-inf.mpg.de/projects/VNect/

Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. 2017. Learning human behaviors from motion capture by adversarial imitation. *CoRR* abs/1707.02201 (2017). arXiv:1707.02201

Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision, ECCV*. 483–499.

Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. 2017. Coarse-to-Fine Volumetric Prediction for Single-Image 3D Human Pose. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Transactions on Graphics (Proc. SIGGRAPH 2018 - to appear)* 37, 4 (2018).

Xue Bin Peng, Glen Berseth, and Michiel van de Panne. 2015. Dynamic Terrain Traversal Skills Using Reinforcement Learning. *ACM Trans. Graph.* 34, 4, Article 80 (July 2015), 11 pages.

Xue Bin Peng, Glen Berseth, and Michiel van de Panne. 2016. Terrain-Adaptive Locomotion Skills Using Deep Reinforcement Learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2016)* 35, 4 (2016).

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, John Schulman, Emanuel Todorov, and Sergey Levine. 2017. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. *CoRR* abs/1709.10087 (2017). arXiv:1709.10087

Gregory Rogez and Cordelia Schmid. 2016. MoCap-guided Data Augmentation for 3D Pose Estimation in the Wild. In *Advances in Neural Information Processing Systems, (NIPS)*.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2015a. Trust Region Policy Optimization. *CoRR* abs/1502.05477 (2015). arXiv:1502.05477

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2015b. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *CoRR* abs/1506.02438 (2015). arXiv:1506.02438

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347

Pierre Sermanet, Corey Lynch, Jasmine Hsu, and Sergey Levine. 2017. Time-Contrastive Networks: Self-Supervised Learning from Multi-View Observation. *CoRR* abs/1704.06888 (2017). arXiv:1704.06888 http://arxiv.org/abs/1704.06888

SFU. 2018. SFU Motion Capture Database. (2018). http://mocap.cs.sfu.ca.

R. Sutton, D. Mcallester, S. Singh, and Y. Mansour. 2001. Policy Gradient Methods for Reinforcement Learning with Function Approximation. (2001), 1057–1063 pages.

Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning* (1st ed.). MIT Press, Cambridge, MA, USA.

C. Taylor. 2000. Reconstruction of articulated objects from point correspondences in single uncalibrated image. *Computer Vision and Image Understanding, CVIU* 80, 10 (2000), 349–363.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. 2017. Distral: Robust Multitask Reinforcement Learning. *CoRR* abs/1707.04175 (2017). arXiv:1707.04175

Bugra Tekin, Artem Rozantsev, Vincent Lepetit, and Pascal Fua. 2016. Direct Prediction of 3D Body Poses from Motion Compensated Sequences. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 991–1000.

Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. 2014. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*. 1799–1807.

Alexander Toshev and Christian Szegedy. 2014. DeepPose: Human Pose Estimation via Deep Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 1653–1660.

Hsiao-Yu Tung, Hsiao-Wei Tung, Ersin Yumer, and Katerina Fragkiadaki. 2017. Self-supervised Learning of Motion Capture. In *Advances in Neural Information Processing Systems*. 5242–5252.

Marek Vondrak, Leonid Sigal, Jessica Hodgins, and Odest Jenkins. 2012. Video-based 3D Motion Capture Through Biped Control. *ACM Trans. Graph.* 31, 4, Article 27 (July 2012), 12 pages. https://doi.org/10.1145/2185520.2185523

Kevin Wampler, Zoran Popović, and Jovan Popović. 2014. Generalizing Locomotion Style to New Animals with Inverse Optimal Regression. *ACM Trans. Graph.* 33, 4, Article 49 (July 2014), 11 pages.

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. 2010. Optimizing Walking Controllers for Uncertain Inputs and Environments. In *ACM SIGGRAPH 2010 Papers (SIGGRAPH '10)*. ACM, New York, NY, USA, Article 73, 8 pages. https://doi.org/10.1145/1833349.1778810

Jack M. Wang, Samuel R. Hamner, Scott L. Delp, and Vladlen Koltun. 2012. Optimizing Locomotion Controllers Using Biologically-based Actuators and Objectives. *ACM Trans. Graph.* 31, 4, Article 25 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185521

Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2016. Convolutional Pose Machines. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 4724–4732.

Jungdam Won, Jongho Park, Kwanyu Kim, and Jehee Lee. 2017. How to Train Your Dragon: Example-guided Control of Flapping Flight. *ACM Trans. Graph.* 36, 6, Article 198 (Nov. 2017), 13 pages.

Weipeng Xu, Avishek Chatterjee, Michael Zollhoefer, Helge Rhodin, Dushyant Mehta, Hans-Peter Seidel, and Christian Theobalt. 2018. MonoPerfCap: Human Performance Capture from Monocular Video. *ACM Transactions on Graphics* (2018). http://gvv.mpi-inf.mpg.de/projects/wxu/MonoPerfCap

Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. 2018a. One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning. *CoRR* abs/1802.01557 (2018). arXiv:1802.01557 http://arxiv.org/abs/1802.01557

Wenhao Yu, Greg Turk, and C. Karen Liu. 2018b. Learning Symmetry and Low-energy Locomotion. *CoRR* abs/1801.08093 (2018). arXiv:1801.08093 http://arxiv.org/abs/1801.08093

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

Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang, and Yichen Wei. 2016. Deep Kinematic Pose Regression. In *ECCV Workshop on Geometry Meets Deep Learning*. 186–201.

X. Zhou, M. Zhu, S. Leonardos, K. Derpanis, and K. Daniilidis. 2015. Sparse Representation for 3D Shape Estimation: A Convex Relaxation Approach. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 4447–4455.

X. Zhou, M. Zhu, S. Leonardos, K. Derpanis, and K. Daniilidis. 2016. Sparseness Meets Deepness: 3D Human Pose Estimation from Monocular Video. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 4966–4975.

SUPPLEMENTARY MATERIAL

Table 4. Summary of Notations

Notation	Definition
$\hat{x}_t$	2D joint prediction at frame t
$\hat{q}_t$	3D pose prediction at frame t
z_t	HMR embedding of the image frame at t
$q(z_t)$	3D pose as a function of the embedding
$F_j(\cdot)$	Forward kinematics function that computes the 3D position of joint j
q_t^*	Final 3D reference pose after motion reconstruction
s_t	state of the simulated character at timestep t
a_t	action
r_t	reward
R_t	return starting at timestep t, $\sum_{l=0}^{T-t} \gamma^l r_{t+l}$
$\mathcal{A}_t$	advantage at timestep t, $R_t - V(s_t)$
τ	a simulated trajectory $(s_0, a_0, s_1, ..., a_{T-1}, s_T)$,
$\pi_\theta(a\|s)$	policy with parameters θ
$\rho_\omega(s_0)$	initial state distribution with parameters ω
α_π	policy stepsize
α_V	value function stepsize
α_ρ	initial state distribution stepsize

A LEARNING ALGORITHM

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

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

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

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

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

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

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

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

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 $\alpha_\pi = 2.5 \times 10^{-6}$ and momentum of 0.9 is used for the policy network, a stepsize of $\alpha_V = 0.01$ is used for the value network, and a stepsize of $\alpha_\rho = 0.001$ is used for the initial state distribution. The discount factor γ is set to 0.95, and λ is set to 0.95 for TD(λ) and GAE(λ). PPO with a clipping threshold of 0.2 is used for policy updates.

ALGORITHM 1: Policy Gradient with Adaptive State Initialization

```
1:  $\theta \leftarrow$ initialize policy parameters
2:  $\psi \leftarrow$ initialize value function parameters
3:  $\omega \leftarrow$ initialize initial state distribution parameters

4:  while not done do
5:      $s_0 \leftarrow$ sample initial state from $\rho_\omega(s_0)$
6:      Simulate rollout $(s_0, a_0, r_0, s_1, \ldots, s_{T+1})$ starting in state $s_0$

7:      for each timestep $t$ do
8:          $R_t \leftarrow$ compute return using $V_\psi$ and TD($\lambda$)
9:          $\mathcal{A}_t \leftarrow$ compute advantage using $V_\psi$ and GAE($\lambda$)
10:     end for

11:     Update value function:
12:     $\psi \leftarrow \psi - \alpha_V \left( \frac{1}{T} \sum_t \nabla_\psi V_\psi(s_t)(R_t - V(s_t)) \right)$

13:     Update policy:
14:     $\theta \leftarrow \theta + \alpha_\pi \left( \frac{1}{T} \sum_t \nabla_\theta \log(\pi_\theta(a_t|s_t)) \mathcal{A}_t \right)$

15:     Update initial state distribution:
16:     $\omega \leftarrow \omega + \alpha_\rho \nabla_\omega \log(\rho_\omega(s_0)) R_0$
17: end while
```

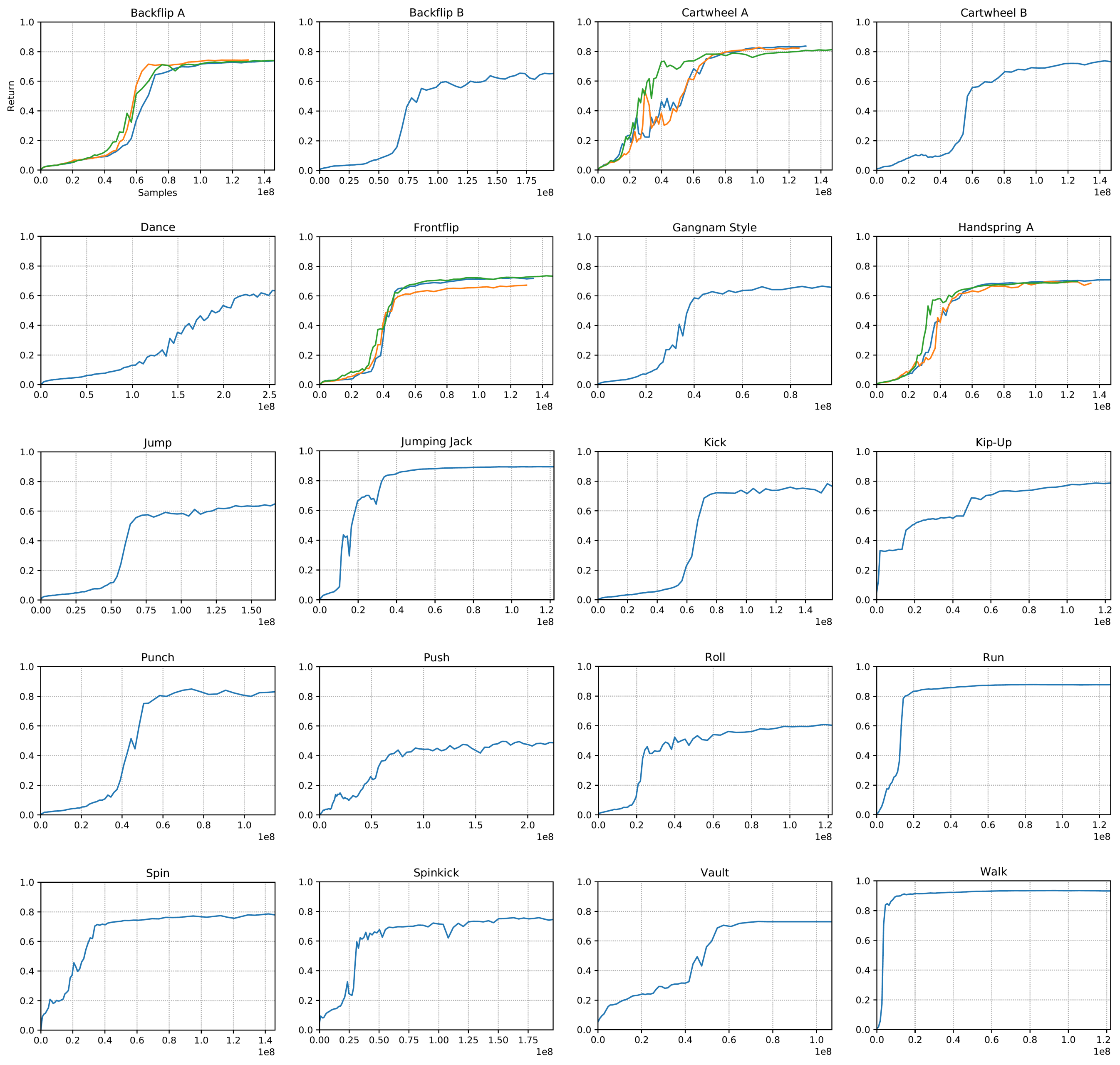


Fig. 15. Learning curves of policies trained for the humanoid. Performance is recorded as the average normalized return over multiple episodes, with 0 representing the minimum possible return and 1 being the maximum.

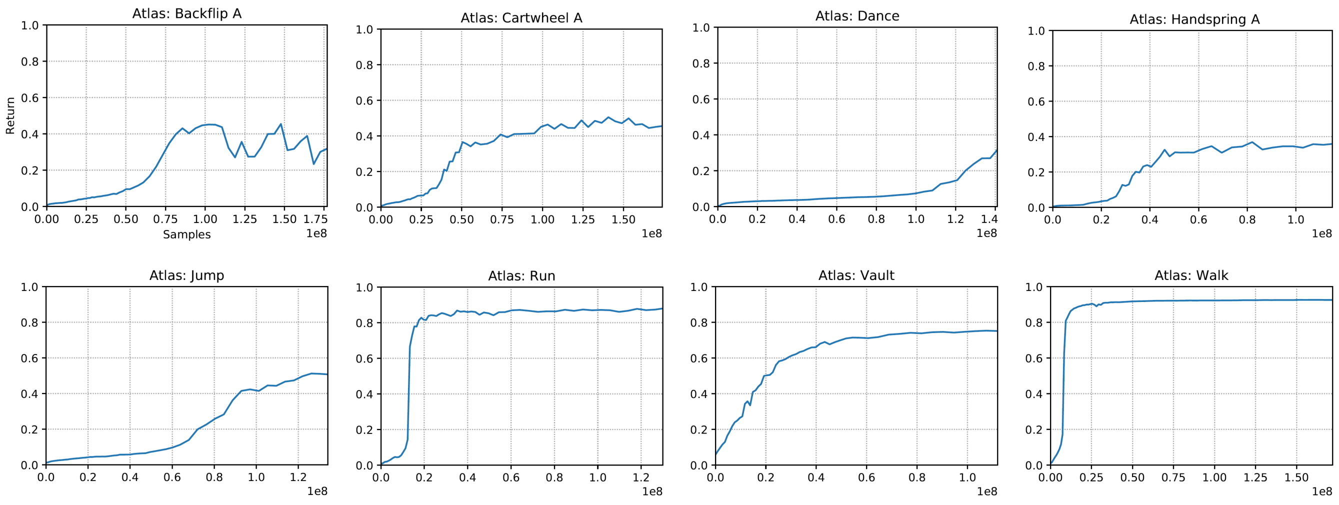


Fig. 16. Learning curves of policies trained for the Atlas.