AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control

XUE BIN PENG*, University of California, Berkeley, USA
ZE MA*, Shanghai Jiao Tong University, China
PIETER ABBEEL, University of California, Berkeley, USA
SERGEY LEVINE, University of California, Berkeley, USA
ANGJOO KANAZAWA, University of California, Berkeley, USA

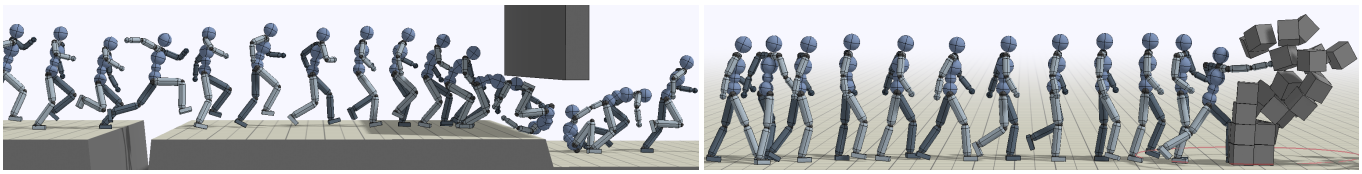


Fig. 1. Our framework enables physically simulated character to solve challenging tasks while adopting stylistic behaviors specified by unstructured motion data. **Left:** A character learns to traverse an obstacles course using a variety of locomotion skills. **Right:** A character learns to walk to and punch a target.

Synthesizing graceful and life-like behaviors for physically simulated characters has been a fundamental challenge in computer animation. Data-driven methods that leverage motion tracking are a prominent class of techniques for producing high fidelity motions for a wide range of behaviors. However, the effectiveness of these tracking-based methods often hinges on carefully designed objective functions, and when applied to large and diverse motion datasets, these methods require significant additional machinery to select the appropriate motion for the character to track in a given scenario. In this work, we propose to obviate the need to manually design imitation objectives and mechanisms for motion selection by utilizing a fully automated approach based on adversarial imitation learning. High-level task objectives that the character should perform can be specified by relatively simple reward functions, while the low-level style of the character's behaviors can be specified by a dataset of unstructured motion clips, without any explicit clip selection or sequencing. For example, a character traversing an obstacle course might utilize a task-reward that only considers forward progress, while the dataset contains clips of relevant behaviors such as running, jumping, and rolling. These motion clips are used to train an adversarial motion prior, which specifies style-rewards for training the character through reinforcement learning (RL). The adversarial RL procedure automatically selects which motion to perform, dynamically interpolating and generalizing from the dataset. Our system produces high-quality motions that are comparable to those achieved by state-of-the-art tracking-based techniques, while also being able to easily accommodate large datasets of unstructured motion clips. Composition of disparate skills emerges automatically from the motion prior, without requiring a high-level motion planner or other task-specific annotations of the motion clips. We demonstrate the effectiveness of our framework on a diverse cast of complex simulated characters and a challenging suite of motor control tasks.

CCS Concepts: • **Computing methodologies** → **Procedural animation**; **Adversarial learning**; *Control methods*.

Additional Key Words and Phrases: character animation, reinforcement learning, adversarial imitation learning

ACM Reference Format:
Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control. *ACM Trans. Graph.* 40, 4, Article 144 (August 2021), 20 pages. https://doi.org/10.1145/3450626.3459670

1 INTRODUCTION

Synthesizing natural and life-like motions for virtual characters is a crucial element for breathing life into immersive experiences, such as films and games. The demand for realistic motions becomes even more apparent for VR applications, where users are provided with rich modalities through which to interact with virtual agents. Developing control strategies that are able to replicate the properties of naturalistic behaviors is also of interest for robotic systems, as natural motions implicitly encode important properties, such as safety and energy efficiency, which are vital for effective operation of robots in the real world. While examples of natural motions are commonplace, identifying the underlying characteristics that constitute these behaviors is nonetheless challenging, and more difficult still to replicate in a controller.

So what are the characteristics that constitute natural and life-like behaviors? Devising quantitative metrics of the *naturalness* of motions has been a fundamental challenge for optimization-based

*Joint first authors.

Authors' addresses: Xue Bin Peng, University of California, Berkeley, 2121 Berkeley Way, Berkeley, CA, 94704, USA, xbpeng@berkeley.edu; Ze Ma, Shanghai Jiao Tong University, 800 Dongchuan Rd, Shanghai, 200240, China, maze1234556@sjtu.edu.cn; Pieter Abbeel, University of California, Berkeley, 2121 Berkeley Way, Berkeley, CA, 94704, USA, pabbeel@cs.berkeley.edu; Sergey Levine, University of California, Berkeley, 2121 Berkeley Way, Berkeley, CA, 94704, USA, svlevine@eecs.berkeley.edu; Angjoo Kanazawa, University of California, Berkeley, 2121 Berkeley Way, Berkeley, CA, 94704, USA, kanazawa@eecs.berkeley.edu.

0730-0301/2021/8-ART144 $15.00
https://doi.org/10.1145/3450626.3459670

character animation techniques [Al Borno et al. 2013; Wampler et al. 2014; Wang et al. 2009]. Heuristics such as symmetry, stability, and effort minimization can improve the realism of motions produced by physically simulated characters [Grochow et al. 2004; Mordatch et al. 2012, 2013; Yu et al. 2018]. But these strategies may not be broadly applicable to all behaviors of interest. Effective applications of these heuristics often require careful balancing of the various objectives, a tuning process that may need to be repeated for each task. Data-driven methods are able to mitigate some of these challenges by leveraging motion clips recorded from real-life actors to guide the behaviors of simulated characters [Da Silva et al. 2008; Liu et al. 2010; Muico et al. 2009; Sok et al. 2007]. A common instantiation of this approach is to utilize a tracking objective that encourages a character to follow particular reference trajectories relevant for a given task. These tracking-based methods can produce high-quality motions for a large repertoire skills. But extending these techniques to effectively leverage large unstructured motion datasets remains challenging, since a suitable motion clip needs to be selected for the character to track at each time step. This selection process is typically performed by a motion planner, which generates reference trajectories for solving a particular task [Bergamin et al. 2019; Park et al. 2019; Peng et al. 2017]. However, constructing an effective motion planner can itself be a challenging endeavour, and entails significant overhead to annotate and organize the motion clips for a desired task. For many applications, it is not imperative to exactly track a particular reference motion. Since a dataset typically provides only a limited collection of example motions, a character will inevitably need to deviate from the reference motions in order to effectively perform a given task. Therefore, the intent is often not for the character to closely track a particular motion, but to adopt general behavioral characteristics depicted in the dataset. We refer to these behavioral characteristics as a *style*.

In this work, we aim to develop a system where users can specify high-level task objectives for a character to perform, while the low-level *style* of a character's movements can be controlled through examples provided in the form of unstructured motion clips. To control the style of a character's motions, we propose adversarial motion priors (AMP), a method for imitating behaviors from raw motion clips without requiring any task-specific annotations or organization of the dataset. Given a set of reference motions that constitutes a desired motion style, the motion prior is modeled as an adversarial discriminator, trained to differentiate between behaviors depicted in the dataset from those produced by the character. The motion prior therefore acts as a general measure of similarity between the motions produced by a character and the motions in the dataset. By incorporating the motion prior in a goal-conditioned reinforcement learning framework, our system is able to train physically simulated characters to perform challenging tasks with natural and life-like behaviors. Composition of diverse behaviors emerges automatically from the motion prior, without the need for a motion planner or other mechanism for selecting *which* clip to imitate.

The central contribution of this work is an adversarial learning approach for physics-based character animation that combines goal-conditioned reinforcement with an adversarial motion prior, which enables the *style* of a character's movements to be controlled via example motion clips, while the *task* is specified through a simple reward function. We present one of the first adversarial learning systems that is able to produce high-quality full-body motions for physically simulated characters. By combining the motion prior with additional task objectives, our system provides a convenient interface through which users can specify high-level directions for controlling a character's behaviors. These task objectives allow our characters to acquire more complex skills than those demonstrated in the original motion clips. While our system is built on well-known adversarial imitation learning techniques, we propose a number of important design decisions that lead to substantially higher quality results than those achieved by prior work, enabling our characters to learn highly dynamic and diverse motors skills from unstructured motion data.

2 RELATED WORK

Developing systems that can synthesize natural motions for virtual characters is one of the fundamental challenges of computer animation. These procedural animation techniques can be broadly categorized as *kinematic methods* and *physics-based methods*. Kinematic methods generally do not explicitly utilize the equations of motion for motion synthesis. Instead, these methods often leverage datasets of motion clips to generate motions for a character [Lee et al. 2002, 2010b]. Given a motion dataset, controllers can be constructed to select an appropriate motion clip to play back for a particular scenario [Agrawal and van de Panne 2016; Safonova and Hodgins 2007; Treuille et al. 2007]. Data-driven methods using generative models, such as Gaussian processes [Levine et al. 2012; Ye and Liu 2010] and neural networks [Holden et al. 2017; Ling et al. 2020; Zhang et al. 2018], have also been applied to synthesize motions online. When provided with sufficiently large and high-quality datasets, kinematic methods are able to produce realistic motions for a large variety of sophisticated skills [Agrawal and van de Panne 2016; Lee et al. 2018, 2010b; Levine et al. 2011; Starke et al. 2019]. However, their ability to synthesize motions for novel situations can be limited by the availability of data. For complex tasks and environments, it can be difficult to collect a sufficient amount of data to cover all possible behaviors that a character may need to perform. This is particularly challenging for nonhuman and fictional creatures, where motion data can be scarce. In this work, we combine data-driven techniques with physics-based animation methods to develop characters that produce realistic and responsive behaviors to novel tasks and environments.

Physics-Based Methods: Physics-based methods address some of the limitations of kinematic methods by synthesizing motions from first principles. These methods typically leverage a physics simulation, or more general knowledge of the equations of motion, to generate motions for a character [Raibert and Hodgins 1991; Wampler et al. 2014]. Optimization techniques, such as trajectory optimization and reinforcement learning, play a pivotal role in many physics-based methods, where controllers that drive a character's motions are produced by optimizing an objective function [Mordatch et al. 2012; Tan et al. 2014; van de Panne et al. 1994]. While these methods are able to synthesize physically plausible motions for novel scenarios, even in the absence of motion data, designing effective objectives that lead to natural behaviors can be exceptionally

difficult. Heuristics derived from prior knowledge of the characteristics of natural motions are commonly included into the objective function, such as symmetry, stability, effort minimization, and many more [Mordatch et al. 2012; Wang et al. 2009; Yu et al. 2018]. Simulating more biologically accurate actuators can also improve motion quality [Geijtenbeek et al. 2013; Jiang et al. 2019; Wang et al. 2012], but may nonetheless yield unnatural behaviors.

Imitation Learning: The challenges of designing objective functions that lead to natural motions have spurred the adoption of data-driven physics-based animation techniques [Da Silva et al. 2008; Kwon and Hodgins 2017; Lee et al. 2010a; Sharon and van de Panne 2005; Zordan and Hodgins 2002], which utilizes reference motion data to improve motion quality. Reference motions are typically incorporated through an imitation objective that encourages a character to imitate motions in the dataset. The imitation objective is commonly implemented as a tracking objective, which attempts to minimize the pose error between the simulated character and target poses from a reference motion [Lee et al. 2010a; Liu et al. 2016, 2010; Peng et al. 2018a; Sok et al. 2007]. Since the pose error is generally computed with respect to a single target pose at a time, some care is required to select an appropriate target pose from the dataset. A simple strategy is to synchronize the simulated character with a given reference motion using a phase variable [Lee et al. 2019; Peng et al. 2018a,b], which is provided as an additional input to the controller. The target pose at each time step can then be conveniently determined by selecting the target pose according to the phase. This strategy has been effective for imitating individual motion clips, but it can be difficult to scale to datasets containing multiple disparate motions, as it may not be possible to synchronize and align multiple reference motions according to a single phase variable. Recent methods have extended these tracking-based techniques to larger motion datasets by explicitly providing target poses from the reference motion that is being tracked as inputs to the controller [Bergamin et al. 2019; Chentanez et al. 2018; Park et al. 2019; Won et al. 2020]. This then allows a controller to imitate different motions depending on the input target poses. However, selecting the appropriate motion for a character to imitate in a given scenario can still entail significant algorithmic overhead. These methods often require a high-level motion planner that selects which motion clip the character should imitate for a given task [Bergamin et al. 2019; Park et al. 2019; Peng et al. 2017]. The character's performance on a particular task can therefore be constrained by the performance of the motion planner.

Another major limitation of tracking-based imitation techniques is the need for a pose error metric when computing the tracking objective [Liu et al. 2010; Peng et al. 2018a; Sharon and van de Panne 2005]. These error metrics are often manually-designed, and it can be challenging to construct and tune a common metric that is effective across all skills that a character is to imitate. Adversarial imitation learning provides an appealing alternative [Abbeel and Ng 2004; Ho and Ermon 2016; Ziebart et al. 2008], where instead of using a manually-designed imitation objective, these algorithms train an adversarial discriminator to differentiate between behaviors generated by an agent from behaviors depicted in the demonstration data (e.g. reference motions). The discriminator then serves as the objective function for training a control policy to imitate the demonstrations. While these methods have shown promising results for motion imitation tasks [Merel et al. 2017; Wang et al. 2017], adversarial learning algorithms can be notoriously unstable and the resulting motion quality still falls well behind what has been achieved with state-of-the-art tracking-based techniques. Peng et al. [2019b] was able to able to produce substantially more realistic motions by regularizing the discriminator with an information bottleneck. However, their method still requires a phase variable to synchronize the policy and discriminator with the reference motion. Therefore, their results are limited to imitating a single motion per policy, and thus not suitable for learning from large diverse motion datasets. In this work, we propose an adversarial method for learning general motion priors from large unstructured datasets that contain diverse motion clips. Our approach does not necessitate any synchronization between the policy and reference motion. Furthermore, our approach does not require a motion planner, or any task-specific annotation and segmentation of the motion clips [Bergamin et al. 2019; Park et al. 2019; Peng et al. 2017]. Instead, composition of multiple motions in furtherance of a task objective emerges automatically through the motion prior. We also present a number of design decisions for stabilizing the adversarial training process, leading to consistent and high quality results.

Latent Space Models: Latent space models can also act as a form of motion prior that leads to more life-like behaviors. These models specify controls through a learned latent representation, which is then mapped to controls for the underlying system [Burgard et al. 2008; Florensa et al. 2017; Hausman et al. 2018; Heess et al. 2016]. The latent representation is typically learned through a pre-training phase using supervised learning or reinforcement learning techniques to encode a diverse range of behaviors into a latent representation. Once trained, this latent representation can be used to build a control hierarchy, where the latent space model acts as a low-level controller, and a separate high-level controller is trained to specify controls via the latent space [Florensa et al. 2017; Haarnoja et al. 2018; Lynch et al. 2020]. For motion control of simulated characters, the latent representation can be trained to encode behaviors from reference motion clips, which then constrains the behavior of a character to be similar to those observed in the motion data, therefore leading to more natural behaviors for downstream tasks [Merel et al. 2019; Peng et al. 2019a]. However, since the realism of the character's motions is enforced implicitly through the latent representation, rather than explicitly through an objective function, it is still possible for the high-level controller to specify latent encodings that produce unnatural behaviors [Merel et al. 2020; Peng et al. 2019a]. Luo et al. [2020] proposed an adversarial domain confusion loss to prevent the high-level controller from specifying encodings that are different from those observed during pre-training. However, since this adversarial objective is applied in the latent space, rather than on the actual motions produced by the character, the model is nonetheless prone to generating unnatural behaviors. Our proposed motion prior directly enforces similarity between the motions produced by the character and those in the reference motion dataset, which enables our method to produce higher fidelity motions than what has been demonstrated by latent space models. Our motion

prior also does not require a separate pre-training phase, and instead, can be trained jointly with the policy.

3 OVERVIEW

Given a dataset of reference motions and a task objective defined by a reward function, our system synthesizes a control policy that enables a character to achieve the task objective in a physically simulated environment, while utilizing behaviors that resemble the motions in the dataset. Crucially, the character's behaviors need not exactly match any specific motion in the dataset, instead its movements need only to adopt more general characteristics exhibited by the corpus of reference motions. These reference motions collectively provide an example-based definition of a behavioral *style*, and by providing the system with different motion datasets, the character can then be trained to perform a task in a variety of distinct styles.

Figure 2 provides a schematic overview of the system. The motion dataset $\mathcal{M}$ consists of a collection of reference motions, where each motion $\mathrm{m}^i = \{\hat{\mathrm{q}}_t^i\}$ is represented as a sequence of poses $\hat{\mathrm{q}}_t^i$. The motion clips may be collected from the mocap of real-life actors or from artist-authored keyframe animations. Unlike previous frameworks, our system can be applied directly on raw motion data, without requiring task-specific annotations or segmentation of a clip into individual skills. The motion of the simulated character is controlled by a policy $\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g})$ that maps the state of the character s_t and a given goal g to a distribution over actions a_t. The actions from the policy specify target positions for proportional-derivative (PD) controllers positioned at each of the character's joints, which in turn produce control forces that drive the motion of the character. The goal g specifies a task reward function $r_t^G = r^G(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_{t+1}, \mathrm{g})$, which defines high-level objectives for the character to satisfy (e.g. walking in a target direction or punching a target). The style objective $r_t^S = r^S(\mathrm{s}_t, \mathrm{s}_{t+1})$ is specified by an adversarial discriminator, trained to differentiate between motions depicted in the dataset from motions produced by the character. The style objective therefore acts as a task-agnostic motion prior that provides an a-priori estimate of the naturalness or style of a given motion, independent of a specific task. The style objective then encourages the policy to produce motions that resemble behaviors depicted in the dataset.

4 BACKGROUND

Our system combines techniques from goal-conditioned reinforcement learning and generative adversarial imitation learning to train control policies that enable simulated characters to perform challenging tasks in a desired behavioral style. In this section, we provide a brief review of these techniques.

4.1 Goal-Conditioned Reinforcement Learning

Our characters are trained through a goal-conditioned reinforcement learning framework, where an agent interacts with an environment according to a policy π in order to fulfill a given goal $\mathrm{g} \in \mathcal{G}$ sampled according to a goal distribution $\mathrm{g} \sim p(\mathrm{g})$. At each time step t, the agent observes the state $\mathrm{s}_t \in \mathcal{S}$ of the system, then samples an action $\mathrm{a}_t \in \mathcal{A}$ from a policy $\mathrm{a}_t \sim \pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g})$. The agent then applies that action, which results in a new state s_{t+1}, as well as a scalar reward $r_t = r(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_{t+1}, \mathrm{g})$. The agent's objective is to learn

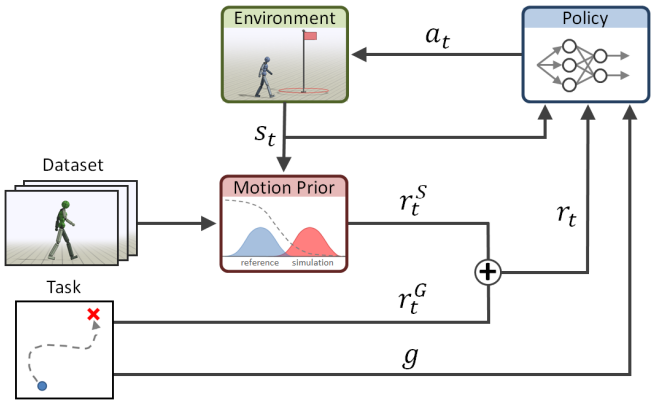


Fig. 2. Schematic overview of the system. Given a motion dataset defining a desired motion style for the character, the system trains a motion prior that specifies style-rewards r_t^S for the policy during training. These style-rewards are combined with task-rewards r_t^G and used to train a policy that enables a simulated character to satisfy task-specific goals g, while also adopting behaviors that resemble the reference motions in the dataset.

a policy that maximizes its expected discounted return $J(\pi)$,

$$J(\pi) = \mathbb{E}_{p(\mathrm{g})}\mathbb{E}_{p(\tau|\pi,\mathrm{g})}\left[\sum_{t=0}^{T-1}\gamma^t r_t\right], \tag{1}$$

where $p(\tau|\pi, \mathrm{g}) = p(\mathrm{s}_0)\prod_{t=0}^{T-1} p(\mathrm{s}_{t+1}|\mathrm{s}_t, \mathrm{a}_t)\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g})$ represents the likelihood of a trajectory $\tau = \{(\mathrm{s}_t, \mathrm{a}_t, r_t)_{t=0}^{T-1}, \mathrm{s}_T\}$ under a policy π for a goal g. $p(\mathrm{s}_0)$ is the initial state distribution, and $p(\mathrm{s}_{t+1}|\mathrm{s}_t, \mathrm{a}_t)$ represents the dynamics of the environment. T denotes the time horizon of a trajectory, and $\gamma \in [0, 1)$ is a discount factor.

4.2 Generative Adversarial Imitation Learning

Generative adversarial imitation learning (GAIL) [Ho and Ermon 2016] adapts techniques developed for generative adversarial networks (GAN) [Goodfellow et al. 2014] to the domain of imitation learning. In the interest of brevity, we exclude the goal g from the notation, but the following discussion readily generalizes to goal-conditioned settings. Given a dataset of demonstrations $\mathcal{M} = \{(\mathrm{s}_i, \mathrm{a}_i)\}$, containing states s_i and actions a_i recorded from an unknown demonstration policy, the objective is to train a policy $\pi(\mathrm{a}|\mathrm{s})$ that imitates the behaviors of the demonstrator. Behavioral cloning can be used to directly fit a policy to map from states observed in $\mathcal{M}$ to their corresponding actions using supervised learning [Bojarski et al. 2016; Pomerleau 1988]. However, if only a small amount of demonstrations are available, then behavioral cloning techniques are prone to drift [Ross et al. 2011]. Furthermore, behavioral cloning is not directly applicable in settings where the demonstration actions are not observable (e.g. reference motion data).

GAIL addresses some of the limitations of behavioral cloning by learning an objective function that measures the similarity between the policy and the demonstrations, and then updating π via reinforcement learning to optimize the learned objective. The objective is modeled as a discriminator $D(\mathrm{s}, \mathrm{a})$, trained to predict whether a given state s and action a is sampled from the demonstrations $\mathcal{M}$

or generated by running the policy π,

$$\arg\min_D \; -\mathbb{E}_{d^{\mathcal{M}}(\mathrm{s},\mathrm{a})}\left[\log\left(D(\mathrm{s},\mathrm{a})\right)\right] - \mathbb{E}_{d^{\pi}(\mathrm{s},\mathrm{a})}\left[\log\left(1-D(\mathrm{s},\mathrm{a})\right)\right]. \tag{2}$$

$d^{\mathcal{M}}(\mathrm{s},\mathrm{a})$ and $d^{\pi}(\mathrm{s},\mathrm{a})$ denote the state-action distribution of the dataset and policy respectively. The policy is then trained using the RL objective detailed in Equation 1, with rewards specified by,

$$r_t = -\log\left(1 - D(\mathrm{s}_t, \mathrm{a}_t)\right). \tag{3}$$

This adversarial training procedure can be interpreted as training a policy to produce states and actions that appear to the discriminator as being indistinguishable from the demonstrations. It can be shown that this objective minimizes the Jensen-Shannon divergence between $d^{\mathcal{M}}(\mathrm{s},\mathrm{a})$ and $d^{\pi}(\mathrm{s},\mathrm{a})$ [Ke et al. 2019; Nowozin et al. 2016].

5 ADVERSARIAL MOTION PRIOR

In this work, we consider reward functions that consist of two components specifying: 1) *what* task a character should perform, and 2) *how* the character should go about performing that task,

$$r(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_{t+1}, \mathrm{g}) = w^G r^G(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_t, \mathrm{g}) + w^S r^S(\mathrm{s}_t, \mathrm{s}_{t+1}). \tag{4}$$

The *what* is represented by a task-specific reward $r^G(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_t, \mathrm{g})$, which defines high-level objectives that a character should satisfy (e.g. moving to a target location). The *how* is represented through a learned task-agnostic style-reward $r^S(\mathrm{s}_t, \mathrm{s}_{t+1})$, which specifies low-level details of the behaviors that the character should adopt when performing the task (e.g., walking vs. running to a target). The two reward terms are combined linearly with weights w^G and w^S. The task-reward r^G can be relatively intuitive and simple to design. However, it can be exceptionally difficult to design a style-reward r^S that leads a character to learn naturalistic behaviors, or behaviors that conform to a particular style. Learning effective style objectives will therefore be the primary focus of this work.

We propose to model the style-reward with a learned discriminator, which we refer to as an adversarial motion prior (AMP), by analogy to the adversarial pose priors that were previously proposed for vision-based pose estimation tasks [Kanazawa et al. 2018]. Unlike standard tracking objectives, which measure pose similarity with respect to a specific reference motion, the motion prior returns a general score indicating the similarity of the character's motion to the motions depicted in the dataset, without explicitly comparing to a particular motion clip. Given a motion dataset, the motion prior is trained using the GAIL framework to predict whether a state transition $(\mathrm{s}_t, \mathrm{s}_{t+1})$ is a *real* sample from the dataset or a *fake* sample produced by the character. The motion prior is independent of the task-specific goal g, therefore a single motion prior can be applied to multiple tasks, and different motion priors can be applied to train policies that perform the same task but in different styles. By combining GAIL with additional task objectives, our approach decouples task specification from style specification, thereby enabling our characters to perform tasks that may not be depicted in the original demonstrations. However, adversarial RL techniques are known to be highly unstable. In the following sections, we discuss a number of design decisions to stabilize the training process and produce higher fidelity results.

5.1 Imitation from Observations

The original formulation of GAIL requires access to the demonstrator's actions [Ho and Ermon 2016]. However, when the demonstrations are provided in the form of motion clips, the actions taken by the demonstrator are unknown, and only states are observed in the data. To extend GAIL to settings with state-only demonstrations, the discriminator can be trained on state transitions $D(\mathrm{s},\mathrm{s}')$ instead of state-action pairs $D(\mathrm{s},\mathrm{a})$ [Torabi et al. 2018],

$$\arg\min_D -\mathbb{E}_{d^{\mathcal{M}}(\mathrm{s},\mathrm{s}')}\left[\log\left(D(\mathrm{s},\mathrm{s}')\right)\right] - \mathbb{E}_{d^{\pi}(\mathrm{s},\mathrm{s}')}\left[\log\left(1-D(\mathrm{s},\mathrm{s}')\right)\right]. \tag{5}$$

$d^{\mathcal{M}}(\mathrm{s},\mathrm{s}')$ and $d^{\pi}(\mathrm{s},\mathrm{s}')$ denote the likelihood of observing a state transition from s to s′ in the dataset $\mathcal{M}$ and by following policy π respectively. Note that if the demonstrator is different from the agent (e.g. a human actor), the observed state transitions may not be physically consistent for the agent, and therefore impossible for the agent to perfectly reproduce. Despite this discrepancy, we show that the discriminator still provides an effective objective for imitating a wide range of behaviors.

5.2 Least-Squares Discriminator

The standard GAN objective detailed in Equation 5 typically uses a sigmoid cross-entropy loss function. However, this loss tends to lead to optimization challenges due to vanishing gradients as the sigmoid function saturates, which can hamper training of the policy [Arjovsky et al. 2017]. A myriad of techniques have been proposed to address these optimization challenges [Arjovsky et al. 2017; Berthelot et al. 2017; Gulrajani et al. 2017; Karras et al. 2017; Kodali et al. 2017; Mescheder et al. 2018; Radford et al. 2015; Salimans et al. 2016]. In this work, we adopt the loss function proposed for least-squares GAN (LSGAN) [Mao et al. 2017], which has demonstrated more stable training and higher quality results for image synthesis tasks. The following objective is used to train the discriminator,

$$\arg\min_D \; \mathbb{E}_{d^{\mathcal{M}}(\mathrm{s},\mathrm{s}')}\left[\left(D(\mathrm{s},\mathrm{s}')-1\right)^2\right] + \mathbb{E}_{d^{\pi}(\mathrm{s},\mathrm{s}')}\left[\left(D(\mathrm{s},\mathrm{s}')+1\right)^2\right]. \tag{6}$$

The discriminator is trained by solving a least-squares regression problem to predict a score of 1 for samples from the dataset and −1 for samples recorded from the policy. The reward function for training the policy is then given by

$$r(\mathrm{s}_t, \mathrm{s}_{t+1}) = \max\left[0,\;\; 1 - 0.25\left(D(\mathrm{s}_t, \mathrm{s}_{t+1}) - 1\right)^2\right]. \tag{7}$$

The additional offset, scaling, and clipping are applied to bound the reward between $[0, 1]$, as is common practice in previous RL frameworks [Peng et al. 2018a, 2016; Tassa et al. 2018]. Mao et al. [2017] showed that this least-squares objective minimizes the Pearson χ^2 divergence between $d^{\mathcal{M}}(\mathrm{s},\mathrm{s}')$ and $d^{\pi}(\mathrm{s},\mathrm{s}')$.

5.3 Discriminator Observations

Since the discriminator specifies rewards for training the policy, selecting an appropriate set of features for use by the discriminator when making its predictions is vital to provide the policy with effective feedback. As such, before a state transition is provided as input to the discriminator, we first apply an observation map $\Phi(\mathrm{s}_t)$ that

extracts a set of features relevant for determining the characteristics of a given motion. The resulting features are then used as inputs to the discriminator $D(\Phi(\mathrm{s}), \Phi(\mathrm{s}'))$. The set of features include:

- Linear velocity and angular velocity of the root, represented in the character's local coordinate frame.
- Local rotation of each joint.
- Local velocity of each joint.
- 3D positions of the end-effectors (e.g. hands and feet), represented in the character's local coordinate frame.

The root is designated to be the character's pelvis. The character's local coordinate frame is defined with the origin located at the root, the x-axis oriented along the root link's facing direction, and the y-axis aligned with the global up vector. The 3D rotation of each spherical joint is encoded using two 3D vectors corresponding to the normal and tangent in the coordinate frame. This rotation encoding provides a smooth and unique representation of a given rotation. This set of observation features for the discriminator is selected to provide a compact representation of the motion across a single state transition. The observations also do not include any task-specific features, thus enabling the motion prior to be trained without requiring task-specific annotation of the reference motions, and allowing motion priors trained with the same dataset to be used for different tasks.

5.4 Gradient Penalty

The interplay between the discriminator and generator in a GAN often results in unstable training dynamics. One source of instability is due to function approximation errors in the discriminator, where the discriminator may assign nonzero gradients on the manifold of real data samples [Mescheder et al. 2018]. These nonzero gradients can cause the generator to *overshoot* and move off the data manifold, instead of converging to the manifold, leading to oscillations and instability during training. To mitigate this phenomenon, a gradient penalty can be applied to penalize nonzero gradients on samples from the dataset [Gulrajani et al. 2017; Kodali et al. 2017; Mescheder et al. 2018]. We incorporate this technique to improve training stability. The discriminator objective is then given by:

$$\begin{aligned}\arg\min_D \quad & \mathbb{E}_{d^{\mathcal{M}}(\mathrm{s},\mathrm{s}')}\left[\left(D(\Phi(\mathrm{s}),\Phi(\mathrm{s}'))-1\right)^2\right] \\ & + \mathbb{E}_{d^{\pi}(\mathrm{s},\mathrm{s}')}\left[\left(D\left(\Phi(\mathrm{s}),\Phi(\mathrm{s}')\right)+1\right)^2\right] \\ & + \frac{w^{\mathrm{gp}}}{2}\,\mathbb{E}_{d^{\mathcal{M}}(\mathrm{s},\mathrm{s}')}\left[\left\|\nabla_{\phi} D(\phi)\Big|_{\phi=(\Phi(\mathrm{s}),\Phi(\mathrm{s}'))}\right\|^2\right], \quad (8)\end{aligned}$$

where w^{gp} is a manually-specified coefficient. Note, the gradient penalty is calculated with respect to the observation features $\phi = (\Phi(\mathrm{s}), \Phi(\mathrm{s}'))$, not the full set of state features $(\mathrm{s}, \mathrm{s}')$. As we show in our experiments, the gradient penalty is crucial for stable training and effective performance.

6 MODEL REPRESENTATION

Given a high-level task objective and a dataset of reference motions, the agent is responsible for learning a control policy that fulfills the task objectives, while utilizing behaviors that resemble the motions depicted in the dataset. In this section, we detail the design of various components of the learning framework.

ALGORITHM 1: Training with AMP

input $\mathcal{M}$: dataset of reference motions
$D \leftarrow$ initialize discriminator
$\pi \leftarrow$ initialize policy
$V \leftarrow$ initialize value function
$\mathcal{B} \leftarrow \emptyset$ initialize reply buffer

while not done **do**
  for trajectory $i = 1, ..., m$ **do**
    $\tau^i \leftarrow \{(\mathrm{s}_t, \mathrm{a}_t, r_t^G)_{t=0}^{T-1}, \mathrm{s}_T^G, \mathrm{g}\}$ collect trajectory with π
    for time step $t = 0, ..., T-1$ **do**
      $d_t \leftarrow D(\Phi(\mathrm{s}_t), \Phi(\mathrm{s}_{t+1}))$
      $r_t^S \leftarrow$ calculate style reward according to Equation 7 using d_t
      $r_t \leftarrow w^G r_t^G + w^S r_t^S$
      record r_t in τ^i
    end for
    store τ^i in $\mathcal{B}$
  end for

  for update step = $1, ..., n$ **do**
    $b^{\mathcal{M}} \leftarrow$ sample batch of K transitions $\{(\mathrm{s}_j, \mathrm{s}_j')\}_{j=1}^K$ from $\mathcal{M}$
    $b^{\pi} \leftarrow$ sample batch of K transitions $\{(\mathrm{s}_j, \mathrm{s}_j')\}_{j=1}^K$ from $\mathcal{B}$
    update D according to Equation 8 using $b^{\mathcal{M}}$ and b^{π}
  end for

  update V and π using data from trajectories $\{\tau^i\}_{i=1}^m$
end while

6.1 States and Actions

The state s_t consists of a set of features that describes the configuration of the character's body. The features are similar to those used by Peng et al. [2018a], which include the relative positions of each link with respective to the root, the rotation of each link as represented using the 6D normal-tangent encoding, along with the link's linear and angular velocities. All features are recorded in the character's local coordinate system. Unlike previous systems, which synchronize the policy with a particular reference motion by including additional phase information in the state, such as scalar phase variables [Lee et al. 2019; Peng et al. 2018a] or target poses [Bergamin et al. 2019; Chentanez et al. 2018; Won et al. 2020], our policies are not trained to explicitly imitate any specific motion from the dataset. Therefore, no such synchronization or phase information is necessary.

Each action a_t specifies target positions for PD controllers positioned at each of the character's joints. For spherical joints, each target is specified in the form of a 3D exponential map $\mathrm{q} \in \mathbb{R}^3$ [Grassia 1998], where the rotation axis v and rotation angle θ can be determined according to:

$$\mathrm{v} = \frac{\mathrm{q}}{||\mathrm{q}||_2}, \qquad \theta = ||\mathrm{q}||_2. \quad (9)$$

This representation provides a more compact parameterization than the 4D axis-angle or quaternion representations used in prior systems [Peng et al. 2018a; Won et al. 2020], while also avoiding gimbal

lock from parameterizations such as euler angles. Target rotations for revolute joints are specified as 1D rotation angles $q = \theta$.

6.2 Network Architecture

Each policy π is modeled by a neural network that maps a given state s_t and goal g to a Gaussian distribution over actions $\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g}) = \mathcal{N}(\mu(\mathrm{s}_t, \mathrm{g}), \Sigma)$, with an input-dependent mean $\mu(\mathrm{s}_t, \mathrm{g})$ and a fixed diagonal covariance matrix Σ. The mean is specified by a fully-connected network with two hidden layers, consisting of 1024 and 512 ReLU [Nair and Hinton 2010], followed by a linear output layer. The values of the covariance matrix $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, ...)$ are manually-specified and kept fixed over the course of training. The value function $V(\mathrm{s}_t, \mathrm{g})$ and discriminator $D(\mathrm{s}_t, \mathrm{s}_{t+1})$ are modeled by separate networks with a similar architecture as the policy.

6.3 Training

Our policies are trained using a combination of GAIL [Ho and Ermon 2016] and proximal-policy optimization (PPO) [Schulman et al. 2017]. Algorithm 1 provides an overview of the training process. At each time step t, the agent receives a task-reward $r_t^G = r^G(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_{t+1}, \mathrm{g})$ from the environment, it then queries the motion prior for a style-reward $r_t^S = r^S(\mathrm{s}_t, \mathrm{s}_{t+1})$, computed according to Equation 7. The two rewards are combined according to Equation 4 to yield the reward for the particular timstep. Following the approach proposed by Peng et al. [2018a], we incorporate reference state initialization and early termination. Reference state initialization is applied by initializing the character to states sampled randomly from all motion clips in the dataset. Early termination is triggered on most tasks when any part of the character's body, with exception of the feet, makes contact with the ground. This termination criteria is disabled for more contact-rich tasks, such as rolling or getting up after a fall.

Once a batch of data has been collected with the policy, the recorded trajectories are used to update the policy and value function. The value function is updated with target values computed using TD(λ) [Sutton and Barto 1998]. The policy is updated using advantages computed using GAE(λ) [Schulman et al. 2015]. Each trajectory recorded from the policy is also stored in a replay buffer $\mathcal{B}$, containing trajectories from past training iterations. The discriminator is updated according to Equation 8 using minibatches of transitions $(\mathrm{s}, \mathrm{s}')$ sampled from the reference motion data set $\mathcal{M}$ and transitions from the replay buffer $\mathcal{B}$. The replay buffer helps to stabilize training by preventing the discriminator from overfitting to the most recent batch of trajectories from the policy.

7 TASKS

To evaluate AMP's effectiveness for controlling the style of a character's motions, we apply our framework to train complex 3D simulated characters to perform various motion control tasks using different motion styles. The characters include a 34 DoF humanoid, a 59 DoF T-Rex, and a 64 DoF dog. A summary of each task is provided below. Please refer to Appendix A for a more in-depth description of each task and their respective reward functions.

Target Heading: In this task, the character's objective is to move along a target heading direction d^* at a target speed v^*. The goal for the policy is specified as $\mathrm{g}_t = (\tilde{\mathrm{d}}_t^*, v^*)$, with $\tilde{\mathrm{d}}_t^*$ being the target direction in the character's local coordinate frame. The target speed is selected randomly between $v^* \in [1, 5]$m/s. For slower moving styles, such as Zombie and Stealthy, the target speed is fixed at 1m/s.

Target Location: In this task, the character's objective is to move to a target location x^*. The goal $\mathrm{g}_t = \tilde{\mathrm{x}}_t^*$ records the target location in the character's local coordinate frame.

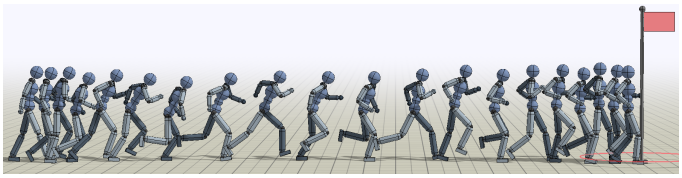
Dribbling: To evaluate our system on more complex object manipulation tasks, we train policies for a dribbling task, where the character's objective is to dribble a soccer ball to a target location. The goal $\mathrm{g}_t = \tilde{\mathrm{x}}_t^*$ records the relative position of the target location with respect to the character. The state s_t is augmented with additional features that describe the state of the ball, including the position $\tilde{\mathrm{x}}_t^{\mathrm{ball}}$, orientation $\tilde{\mathrm{q}}_t^{\mathrm{ball}}$, linear velocity $\tilde{\dot{\mathrm{x}}}_t^{\mathrm{ball}}$, and angular velocity $\tilde{\dot{\mathrm{q}}}_t^{\mathrm{ball}}$ of the ball in the character's local coordinate frame.

Strike: To demonstrate AMP's ability to compose diverse behaviors, we consider a task where the character's objective is to strike a target using a designated end-effector (e.g. hands). The target may be located at various distances from the character. Therefore, the character must first move close to the target before striking it. These distinct phases entail different optimal behaviors, and thus require the policy to compose and transition between the appropriate skills. The goal $\mathrm{g}_t = (\tilde{\mathrm{x}}_t^*, h_t)$ records the location of the target $\tilde{\mathrm{x}}_t^*$ in the character's local coordinate frame, along with an indicator variable h_t that specifies if the target has already been hit.

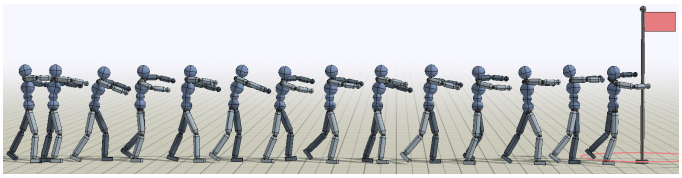
Obstacles: Finally, we consider tasks that involve visual perception and interaction with more complex environments, where the character's objective is to traverse an obstacle-filled terrain, while maintaining a target speed. Policies are trained for two types of environments: 1) An environment containing a combination of obstacles include gaps, steps, and overhead obstructions that the character must duck under. 2) An environment containing narrow stepping stones that requires more precise contact planning. Examples of the environments are available in Figure 1 and 3. In order for the policy to perceive the upcoming obstacles, the state is augmented with a 1D height-field of the upcoming terrain.

8 RESULTS

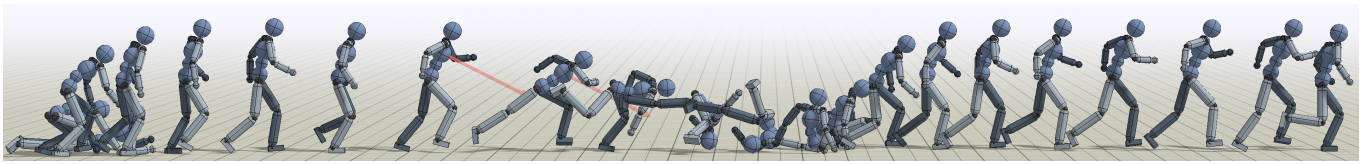
We evaluate our framework's effectiveness on a suite of challenging motion control tasks with complex simulated characters. First, we demonstrate that our approach can readily scale to large unstructured datasets containing diverse motion clips, which then enables our characters to perform challenging tasks in a natural and life-like manner by imitating behaviors from the dataset. The characters automatically learn to compose and generalize different skills from the motion data in order to fulfill high-level task objectives, without requiring mechanisms for explicit motion selection. We then evaluate AMP on a single-clip imitation task, and show that our method is able to closely imitate a diverse corpus of dynamic and acrobatic skills, producing motions that are nearly indistinguishable from reference motions recorded from human actors. Behaviors learned by the characters can be viewed in the supplementary video.



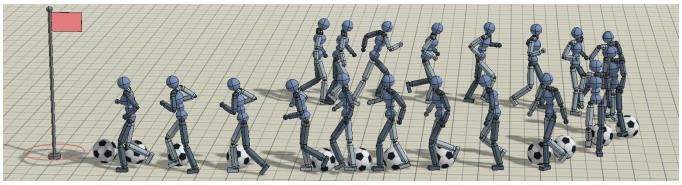
(a) Humanoid: Target Location (Locomotion)



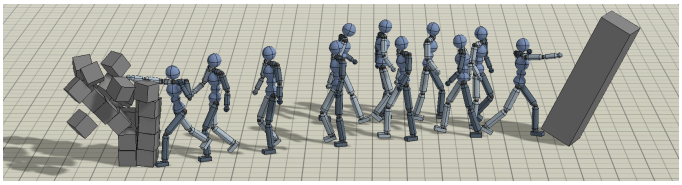
(b) Humanoid: Target Location (Zombie)



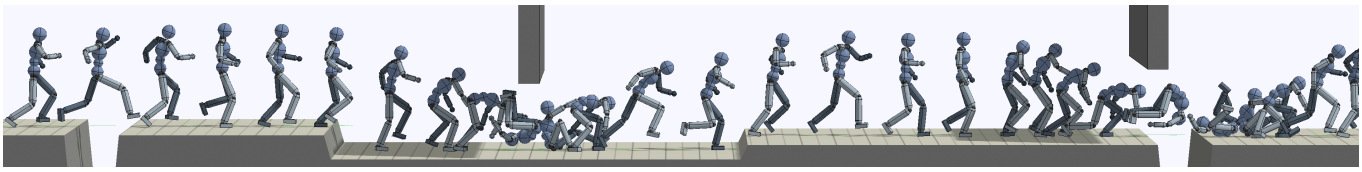
(c) Humanoid: Target Heading (Locomotion + Getup)



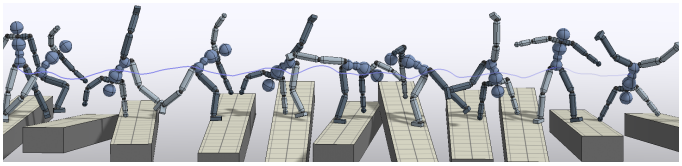
(d) Humanoid: Dribble (Locomotion)



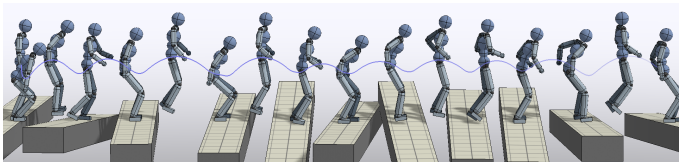
(e) Humanoid: Strike (Walk + Punch)



(f) Humanoid: Obstacles (Run + Leap + Roll)



(g) Humanoid: Stepping Stones (Cartwheel)



(h) Humanoid: Stepping Stones (Jump)

Fig. 3. The motion prior can be trained with large datasets of diverse motions, enabling simulated characters to perform complex tasks by composing a wider range of skills. Each environment is denoted by "Character: Task (Dataset)".

8.1 Experimental Setup

All environments are simulated using the Bullet physics engine [Coumans et al. 2013], with a simulation frequency of 1.2kHz. The policy is queried at 30Hz, and each action specifies target positions for PD controllers positioned at the character's joints. All neural networks are implemented using Tensorflow [Abadi et al. 2015]. The gradient penalty coefficient is set to $w^{\text{gp}} = 10$. Detailed hyperparameter settings are available in Appendix B. Reference motion clips are collected from a combination of public mocap libraries, custom recorded mocap clips, and artist-authored keyframe animations [CMU [n.d.]; SFU [n.d.]; Zhang et al. 2018]. Depending on the task and character, each policy is trained with 100-300 million samples, requiring approximately 30-140 hours on 16 CPU cores. Code for our system will be released upon publication of this paper.

8.2 Tasks

In this section, we demonstrate AMP's effectiveness for controlling the style of a character's motions as it performs other high-level tasks. The weights for the task-reward and style-reward are set to $w^G = 0.5$ and $w^S = 0.5$ for all tasks. The character can be trained to perform tasks in a variety of distinct styles by providing the motion prior with different datasets. Figure 3 illustrates behaviors learned by the Humanoid on various tasks. Table 1 records the performance of the policies with respect to the normalized task return, and summary statistics of the different datasets used to train the motion priors are available in Table 2. AMP can accommodate large unstructured datasets, with the largest dataset containing 56 clips from 8 different human actors, for a total of 434s of motion data. In the case of the *Target Heading* task, a motion prior trained using a locomotion dataset, containing walking, running, and jogging motions, leads to a policy that executes different locomotion gaits depending on the target speed. Transitions between various gaits

Table 1. Performance statistics of combining AMP with additional task objectives. Performance is recorded as the average normalized task return, with 0 being the minimum possible return per episode and 1 being the maximum possible return. The return is averaged across 3 models initialized with different random seeds, with 32 episodes recorded per model. The motion prior can be trained with different datasets to produce policies that adopt distinct stylistic behaviors when performing a particular task.

Character	Task	Dataset	Task Return
Humanoid	Target Heading	Locomotion	0.90 ± 0.01
		Walk	0.46 ± 0.01
		Run	0.63 ± 0.01
		Stealthy	0.89 ± 0.02
		Zombie	0.94 ± 0.00
	Target Location	Locomotion	0.63 ± 0.01
		Zombie	0.50 ± 0.00
	Obstacles	Run + Leap + Roll	0.27 ± 0.10
	Stepping Stones	Cartwheel	0.43 ± 0.03
		Jump	0.56 ± 0.12
	Dribble	Locomotion	0.78 ± 0.05
		Zombie	0.60 ± 0.04
	Strike	Walk + Punch	0.73 ± 0.02
T-Rex	Target Location	Locomotion	0.36 ± 0.03

Table 2. Summary statistics of the different datasets used to train the motion priors. We record the total length of motion clips in each dataset, along with the number of clips, and the number of subjects (e.g. human actors) that the clips were recorded from.

Character	Dataset	Size (s)	Clips	Subjects
Humanoid	Cartwheel	13.6	3	1
	Jump	28.6	10	4
	Locomotion	434.1	56	8
	Run	204.4	47	3
	Run + Leap + Roll	22.1	10	7
	Stealthy	136.5	3	1
	Walk	229.6	9	5
	Walk + Punch	247.8	15	9
	Zombie	18.3	1	1
T-Rex	Locomotion	10.5	5	1

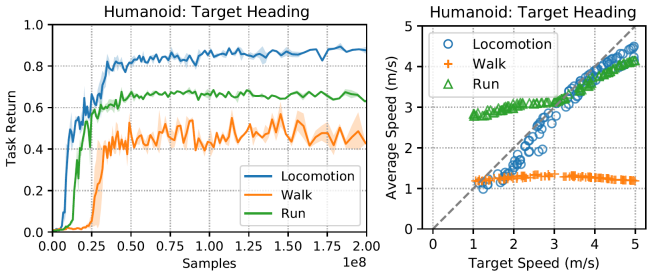


Fig. 4. Performance of Target Heading policies trained with different datasets. **Left:** Learning curves comparing the normalized task returns of policies trained with a large dataset of diverse locomotion clips to policies trained with only walking or running reference motions. Three models are trained using each dataset. **Right:** Comparison of the target speed with the average speed achieved by the different policies. Policies trained using the larger Locomotion dataset is able to more closely follow the various target speeds by imitating different gaits.

emerge automatically through the motion prior, with the character adopting walking gaits at slow speeds (~ 1m/s), switching to jogging gaits at faster speeds (~ 2.5m/s), and breaking into a fast run as the target speed approaches (~ 4.5m/s). The motion prior also leads to other human-like strategies, such as banking into turns, and slowing down before large changes in direction. The policies develop similar behaviors for the *Target Location* task. When the target is near the character, the policy adopts slower walking gaits. But when the target is further away, the character automatically transitions into a run. These intricate behaviors arise naturally from the motion prior, without requiring a motion planner to explicitly select which motion the character should execute in a given scenario, such as those used in prior systems [Bergamin et al. 2019; Luo et al. 2020; Peng et al. 2017]. In addition to standard locomotion gaits, the motion prior can also be trained for more stylistic behaviors, such as walking like a shambling zombie or walking in a stealthy manner. Our framework enables the character to acquire these distinct styles by simply providing the motion prior with different unstructured motion datasets.

To determine whether the transitions between distinct gaits are a product of the motion prior or a result of the task objective, we train policies to perform the *Target Heading* task using limited datasets containing only walking or running data. Figure 4 compares the performance of policies trained with these different datasets. Policies trained with only walking motions learn to perform only walking gaits, and do not show any transitions to faster running gaits even at faster target speeds. As a result, these policies are not able to achieve the faster target speeds. Similarly, policies trained with only running motions are not able to match slower target speeds. Training the motion prior with a diverse dataset results in more flexible and optimal policies that are able to achieve a wider range of target speeds. This indicates that the diversity of behaviors exhibited by our policies can in large part be attributed to the motion prior, and is not solely a result of the task objective.

To further illustrate AMP's ability to compose disparate skills, we introduce additional reference motions into the dataset for getting up from the ground in various configurations. These additional motion clips then enable our character to recover from a fall and continue to perform a given task (Figure 3(c)). The policy also discovers novel recovery behaviors that are not present in the dataset. When the character falls forward, it tucks its body into a roll during the fall in order to more quickly transition into a getup behavior. While this particular behavior is not present in the motion clips, the policy is able to generalize behaviors observed in the dataset to produce novel and naturalistic strategies for new scenarios.

For the *Strike* task (Figure 1), the motion prior is trained using a collection of walking motion clips and punching motion clips. The resulting policy learns to walk to the target when it is far away, and then transition to a punching motion once it is within range to hit the target. Note that the motion clips in the dataset contain strictly walking-only motions or punching-only motion, and none of the clips show an actor walking to and punching a target. Instead, the policy learns to temporally sequence these different behaviors in

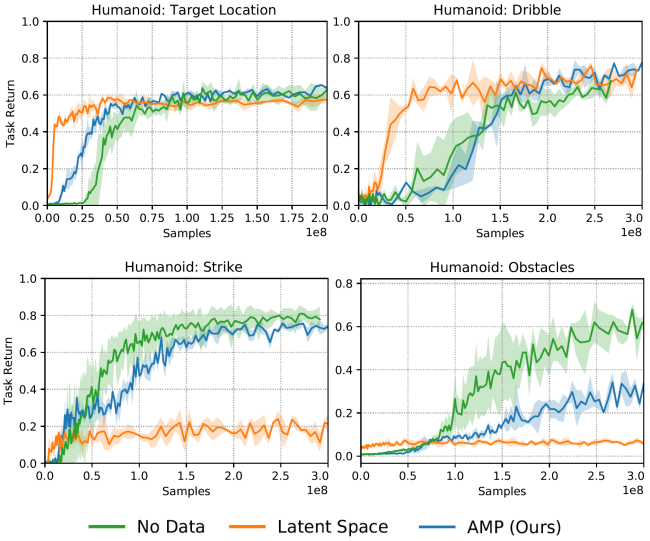


Fig. 5. Learning curves comparing the task performance of AMP to latent space models (Latent Space) and policies trained from scratch without motion data (No Data). Our method achieves comparable performance across the various tasks, while also producing higher fidelity motions.

order to fulfill the high-level task objectives. Again, this composition of different skills emerges automatically from the motion prior, without requiring a motion planner or other mechanisms for motion selection.

Finally, our system can also train visuomotor policies for traversing obstacle-filled environments. By providing the motion prior with a collection of locomotion clips and rolling clips, the character learns to utilize these diverse behaviors to traverse the different obstacles. The character learns to leap over obstacles such as gaps. But as it approaches the overhead obstructions, the character transitions into a rolling behavior in order to pass underneath the obstacles. Previous systems that have demonstrated similar composition of diverse maneuvers for clearing obstacle have typically required a separate motion planner or manual annotations [Liu et al. 2012; Park et al. 2019]. Our approach provides a unified framework where the same underlying algorithm is able to learn how to perform the various skills and which skill to execute in a given scenario. Furthermore, the character can also be trained to traverse obstacles in distinct styles by providing the motion prior with different motion clips, such as jumping or cartwheeling across stepping stones (Figure 3).

8.3 Comparisons

An alternative approach for learning a motion prior from unstructured motion data is to build a latent space model [Heess et al. 2016; Lynch et al. 2020; Merel et al. 2020; Peng et al. 2019a]. Unlike AMP, which encourages a character to adopt a desired motion style directly through an optimization objective, a latent space model enforces a particular motion style indirectly, by using a latent representation to constrain the policy's actions to those that produce motions of the desired style. To compare AMP to these latent space models, we first pre-train a low-level controller using a motion tracking objective to imitate the same set of reference motions that are used

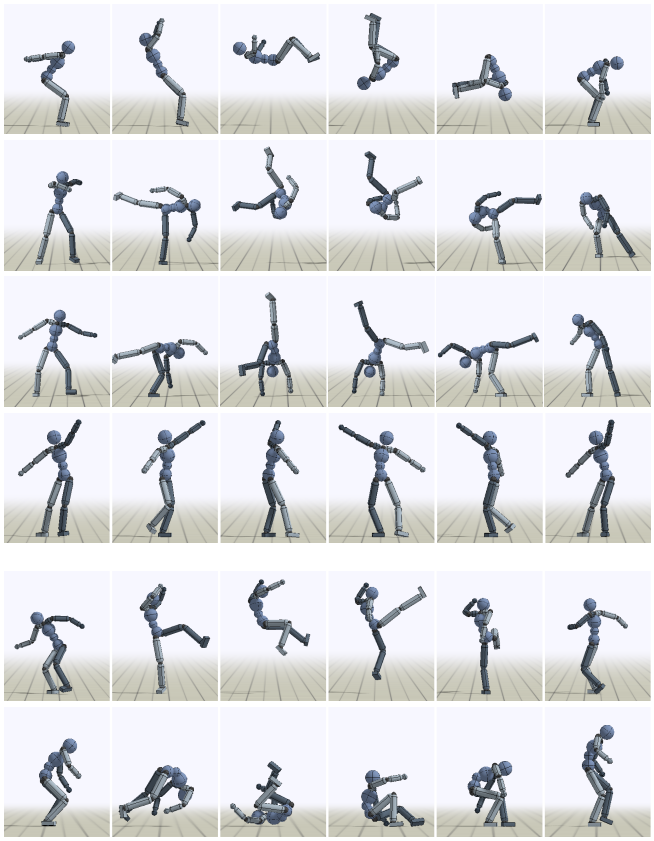
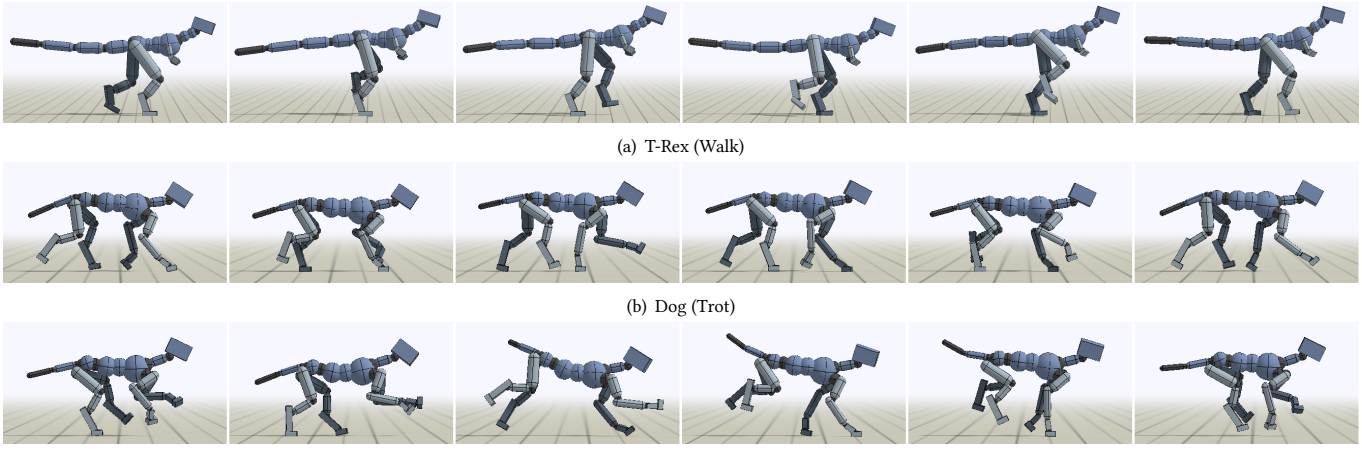


Fig. 6. Snapshots of behaviors learned by the Humanoid on the single-clip imitation tasks. **Top-to-bottom:** back-flip, side-flip, cartwheel, spin, spin-kick, roll. AMP enables the character to closely imitate a diverse corpus of highly dynamic and acrobatic skills.

to train the motion prior. The learned low-level controller is then used to train separate high-level controllers for each downstream task. Note that reference motions are used only during pre-training, and the high-level controllers are trained to optimize only the task objectives. A more in-depth description of the experimental setup for the latent space model is available in Appendix C.

A qualitative comparison of the behaviors learned using AMP and the latent space model is available in the supplementary video. Figure 5 compares the task performance of the different models, along with a baseline model trained from scratch for each task without leveraging any motion data. Both AMP and the latent space models are able to produce substantially more life-like behaviors than the baseline models. For the latent space models, since the low-level and high-level controllers are trained separately, it is possible for the distribution of encodings specified by the high-level controller to be different than the distribution of encodings observed by the low-level controller during pre-training [Luo et al. 2020]. This in turn can result in unnatural motions that deviate from the behaviors depicted in the original dataset. AMP enforces a motion style directly through the reward function, and is therefore able to better mitigate some of these artifacts. The more structured exploration behaviors from the latent space model enable the policies to solve



(a) T-Rex (Walk)

(b) Dog (Trot)

(c) Dog (Canter)

Fig. 7. AMP can be used to train complex non-humanoid characters, such as a 59 DoF T-Rex and a 64 DoF dog. By providing the motion prior with different reference motion clips, the characters can be trained to perform various locomotion gaits, such as trotting and cantering.

downstream tasks more quickly. However, the pre-training stage used to construct the low-level controller can itself be sample intensive. In our experiments, the low-level controllers are trained using 300 million samples before being transferred to downstream tasks. With AMP, no such pre-training is necessary, and the motion prior can be trained jointly with the policy.

8.4 Single-Clip Imitation

Although our goal is to train characters with large motion datasets, to evaluate the effectiveness of our framework for imitating behaviors from motion clips, we consider a single-clip imitation task. In this setting, the character's objective is to imitate a single motion clip at a time, without additional task objectives. Therefore, the policy is trained solely to maximize the style-reward r_t^S from the motion prior. Unlike previous motion tracking methods, our approach does not require a manually designed tracking objective or a phase-based synchronization of the reference motion and the policy Peng et al. [2018a]. Table 3 summarizes the performance of policies trained using AMP to imitate a diverse corpus of motions. Figure 6 and 7 illustrate examples of motions learned by the characters. Performance is evaluated using the average pose error, where the pose error e_t^{pose} at each time step t is computed between the pose of the simulated character and the reference motion using the relative positions of each joint with respect to the root (in units of meters),

$$e_t^{\text{pose}} = \frac{1}{N^{\text{joint}}} \sum_{j \in \text{joints}} \left\| (\mathrm{x}_t^j - \mathrm{x}_t^{\text{root}}) - (\hat{\mathrm{x}}_t^j - \hat{\mathrm{x}}_t^{\text{root}}) \right\|_2 . \quad (10)$$

x_t^j and $\hat{\mathrm{x}}_t^j$ denote the 3D Cartesian position of joint j from the simulated character and the reference motion, and N^{joint} is the total number of joints in the character's body. This method of evaluating motion similarity has previously been reported to better conform to human perception of motion similarity [Harada et al. 2004; Tang et al. 2008]. Since AMP does not use a phase variable to synchronize the policy with the reference motion, the motions may progress at different rates, resulting in de-synchronization that can lead to large pose errors even when the overall motions are similar. To better evaluate the similarity of the motions, we first apply dynamic time warping (DTW) to align the reference motion with the motion of the simulated character [Sakoe and Chiba 1978], before computing the pose error between the two aligned motions. DTW is applied using Equation 10 as the cost function.

AMP is able to closely imitate a large variety of highly dynamic skills, while also avoiding many of the visual artifacts exhibited by prior adversarial motion imitation systems [Merel et al. 2017; Wang et al. 2017]. We compare the performance of our system to results produced by the motion tracking approach from Peng et al. [2018a], which uses a manually designed reward function and requires synchronization of the policy with a reference motion via a phase variable. Figure 8 compares the learning curves of the different methods. Since the tracking-based policies are synchronized with their respective reference motions, they are generally able to learn faster and achieve lower errors than policies trained with AMP. Nonetheless, our method is able to produce results of comparable quality without the need to manually design or tune reward functions for different motions. However, for some motions, such as the Front-Flip, AMP is prone to converging to locally optimal behaviors, where instead of performing a flip, the character learns to simply shuffle forwards in order to avoid falling. Tracking-based methods can mitigate these local optima by terminating an episode early if the character's pose deviates too far from the reference motion [Peng et al. 2018a; Won et al. 2020]. However, this strategy is not directly applicable to AMP, since the policy is not synchronized with the reference motion. But as shown in the previous sections, this lack of synchronization is precisely what allows AMP to easily leverage large datasets of diverse motion clips to solve more complex tasks.

Table 3. Performance statistics of imitating individual motion clips without task objectives. "Dataset Size" records the total length of motion data used for each skill. Performance is recorded as the average pose error (in units of meters) between the time-warped trajectories from the reference motion and simulated character. The pose error is averaged across 3 models initialized with different random seeds, with 32 episodes recorded per model. Each episode has a maximum length of 20s. We compare our method (AMP) with the motion tracking approach proposed by Peng et al. [2018a]. AMP is able to closely imitate a diverse repertoire of complex motions, without manual reward engineering.

Character	Motion	Dataset Size	Motion Tracking	AMP (Ours)
Humanoid	Back-Flip	1.75s	0.076 ± 0.021	0.150 ± 0.028
	Cartwheel	2.72s	0.039 ± 0.011	0.067 ± 0.014
	Crawl	2.93s	0.044 ± 0.001	0.049 ± 0.007
	Dance	1.62s	0.038 ± 0.001	0.055 ± 0.015
	Front-Flip	1.65s	0.278 ± 0.040	0.425 ± 0.010
	Jog	0.83s	0.029 ± 0.001	0.056 ± 0.001
	Jump	1.77s	0.033 ± 0.001	0.083 ± 0.022
	Roll	2.02s	0.072 ± 0.018	0.088 ± 0.008
	Run	0.80s	0.028 ± 0.002	0.075 ± 0.015
	Spin	1.00s	0.063 ± 0.022	0.047 ± 0.002
	Side-Flip	2.44s	0.191 ± 0.043	0.124 ± 0.012
	Spin-Kick	1.28s	0.042 ± 0.001	0.058 ± 0.012
	Walk	1.30s	0.018 ± 0.005	0.030 ± 0.001
	Zombie	1.68s	0.049 ± 0.013	0.058 ± 0.014
T-Rex	Turn	2.13s	0.098 ± 0.011	0.284 ± 0.023
	Walk	2.00s	0.069 ± 0.005	0.096 ± 0.027
Dog	Canter	0.45s	0.026 ± 0.002	0.034 ± 0.002
	Pace	0.63s	0.020 ± 0.001	0.024 ± 0.003
	Spin	0.73s	0.026 ± 0.002	0.086 ± 0.008
	Trot	0.52s	0.019 ± 0.001	0.026 ± 0.001

8.5 Ablations

Our system is able to produce substantially higher fidelity motions than prior adversarial learning frameworks for physics-based character control [Merel et al. 2017; Wang et al. 2017]. In this section, we identify critical design decisions that lead to more stable training and higher quality results. Figure 8 compares learning curves of policies trained on the single-clip imitation tasks with different components of the system disabled. Gradient penalty proves to be the most vital component. Models trained without this regularization tend to exhibit large performance fluctuations over the course of the training, and lead to noticeable visual artifacts in the final policies, as shown in the supplementary video. The addition of the gradient penalty not only improves stability during training, but also leads to substantially faster learning across a large set of skills. The inclusion of velocity features in the discriminator's observations is also an important component for imitating some motions. In principle, including consecutive poses as input to the discriminator should provide some information that can be used to infer the joint velocities. But we found that this was insufficient for some motions, such as rolling. As shown in the supplementary video, in the absence of velocity features, the character is prone to converging to a strategy of holding a fixed pose on the ground, instead of performing a roll. The additional velocity features are able to mitigate these undesirable behaviors.

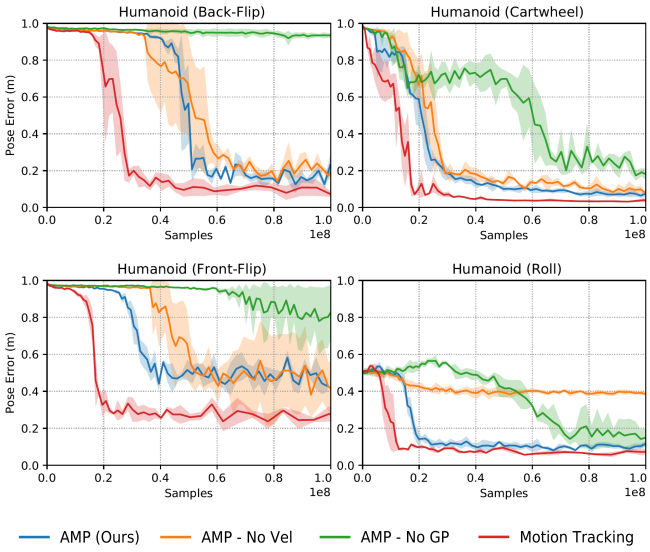


Fig. 8. Learning curves of various methods on the single-clip imitation tasks. We compare AMP to the motion tracking approach proposed by Peng et al. [2018a] (Motion Tracking), as well a version of AMP without velocity features for the discriminator (AMP - No Vel), and AMP without the gradient penalty regularizer (AMP - No GP). A comprehensive collection of learning curves for all skills are available in the Appendix. AMP produces results of comparable quality when compared to prior tracking-based methods, without requiring a manually designed reward function or synchronization between the policy and reference motion. Velocity features and gradient penalty are vital for effective and consistent results on challenging skills.

9 DISCUSSION AND LIMITATIONS

In this work, we presented an adversarial learning system for physics-based character animation that enables characters to imitate diverse behaviors from large unstructured datasets, without the need for motion planners or other mechanisms for clip selection. Our system allows users to specify high-level task objectives for controlling a character's behaviors, while the more granular low-level style of a character's motions can be controlled using a learned motion prior. Composition of disparate skills in furtherance of a task objective emerges automatically from the motion prior. The motion prior also enables our characters to closely imitate a rich repertoire of highly dynamic skills, and produces results that are on par with tracking-based techniques, without requiring manual reward engineering or synchronization between the controller and the reference motions.

Our system demonstrates that adversarial imitation learning techniques can indeed produce high fidelity motions for complex skills. However, like many other GAN-based techniques, AMP is susceptible to mode collapse. When provided with a large dataset of diverse motion clips, the policy is prone to imitating only a small subset of the example behaviors, ignoring other behaviors that may ultimately be more optimal for a given task. The motion priors in our experiments are also trained from scratch for each policy. But since the motion prior is largely task-agnostic, it should in principle be possible to transfer and reuse motion priors for different policies and tasks. Exploring techniques for developing general and transferable motion priors may lead to modular objective functions that

can be conveniently incorporated into downstream tasks, without requiring retraining for each new task. While the motion prior does not require direct access to task-specific information, the data used to train the motion prior is generated by policies trained to perform a particular task. This may introduce some task dependencies into the motion prior, which can hinder its ability to be transferred to other tasks. Training motion priors using data generated from larger and more diverse repertoires of tasks may help to facilitate transferring the learned motion priors to new tasks. Our experiments also focus primarily on tasks that involve temporal composition of different skills, which require the character to perform different behaviors at different points in time. However, spatial composition might also be vital for some tasks that require a character to perform multiple skills simultaneously. Developing motion priors that are more amenable to spatial composition of disparate skills may lead to more flexible and sophisticated behaviors. Despite these limitations, we hope this work provides a useful tool that enables physically simulated characters to take advantage of the large motion datasets that have been so effective for kinematic animation techniques, and open exciting directions for future exploration in data-driven physics-based character animation.

ACKNOWLEDGMENTS

We thank Sony Interactive Entertainment for providing reference motion data for this project, Bonny Ho for narrating the video, the anonymous reviewers for their helpful feedback, and AWS for providing computational resources. This research was funded by an NSERC Postgraduate Scholarship, and a Berkeley Fellowship for Graduate Study.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/ Software available from tensorflow.org.

Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship Learning via Inverse Reinforcement Learning. In *Proceedings of the Twenty-First International Conference on Machine Learning* (Banff, Alberta, Canada) *(ICML '04)*. Association for Computing Machinery, New York, NY, USA, 1. https://doi.org/10.1145/1015330.1015430

Shailen Agrawal and Michiel van de Panne. 2016. Task-based Locomotion. *ACM Transactions on Graphics (Proc. SIGGRAPH 2016)* 35, 4 (2016).

M. Al Borno, M. de Lasa, and A. Hertzmann. 2013. Trajectory Optimization for Full-Body Movements with Complex Contacts. *IEEE Transactions on Visualization and Computer Graphics* 19, 8 (2013), 1405–1414. https://doi.org/10.1109/TVCG.2012.325

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks *(Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, International Convention Centre, Sydney, Australia, 214–223. http://proceedings.mlr.press/v70/arjovsky17a.html

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: Data-Driven Responsive Control of Physics-Based Characters. *ACM Trans. Graph.* 38, 6, Article 206 (Nov. 2019), 11 pages. https://doi.org/10.1145/3355089.3356536

David Berthelot, Tom Schumm, and Luke Metz. 2017. BEGAN: Boundary Equilibrium Generative Adversarial Networks. *CoRR* abs/1703.10717 (2017). arXiv:1703.10717 http://arxiv.org/abs/1703.10717

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to End Learning for Self-Driving Cars. *CoRR* abs/1604.07316 (2016). arXiv:1604.07316 http://arxiv.org/abs/1604.07316

W. Burgard, O. Brock, and C. Stachniss. 2008. *Learning Omnidirectional Path Following Using Dimensionality Reduction*. 257–264.

Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. 2018. Physics-Based Motion Capture Imitation with Deep Reinforcement Learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games* (Limassol, Cyprus) *(MIG '18)*. Association for Computing Machinery, New York, NY, USA, Article 1, 10 pages. https://doi.org/10.1145/3274247.3274506

CMU. [n.d.]. CMU Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu/.

Erwin Coumans et al. 2013. Bullet physics library. *Open source: bulletphysics. org* 15, 49 (2013), 5.

M. Da Silva, Y. Abe, and J. Popovic. 2008. Simulation of Human Motion Data using Short-Horizon Model-Predictive Control. *Computer Graphics Forum* (2008). https://doi.org/10.1111/j.1467-8659.2008.01134.x

Carlos Florensa, Yan Duan, and Pieter Abbeel. 2017. Stochastic Neural Networks for Hierarchical Reinforcement Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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).

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

F. Sebastin Grassia. 1998. Practical Parameterization of Rotations Using the Exponential Map. *J. Graph. Tools* 3, 3 (March 1998), 29–48. https://doi.org/10.1080/10867651.1998.10487493

Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. 2004. Style-Based Inverse Kinematics. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 522–531. https://doi.org/10.1145/1015706.1015755

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5767–5777. http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. 2018. Latent Space Policies for Hierarchical Reinforcement Learning *(Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm Sweden, 1851–1860. http://proceedings.mlr.press/v80/haarnoja18a.html

T. Harada, S. Taoka, T. Mori, and T. Sato. 2004. Quantitative evaluation method for pose and motion similarity based on human perception. In *4th IEEE/RAS International Conference on Humanoid Robots, 2004.*, Vol. 1. 494–512 Vol. 1. https://doi.org/10.1109/ICHR.2004.1442140

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. 2018. Learning an Embedding Space for Transferable Robot Skills. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rk07ZXZRb

Nicolas Heess, Gregory Wayne, Yuval Tassa, Timothy P. Lillicrap, Martin A. Riedmiller, and David Silver. 2016. Learning and Transfer of Modulated Locomotor Controllers. *CoRR* abs/1610.05182 (2016). arXiv:1610.05182 http://arxiv.org/abs/1610.05182

Jonathan Ho and Stefano Ermon. 2016. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 4565–4573. http://papers.nips.cc/paper/6391-generative-adversarial-imitation-learning.pdf

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. https://doi.org/10.1145/3072959.3073663

Yifeng Jiang, Tom Van Wouwe, Friedl De Groote, and C. Karen Liu. 2019. Synthesis of Biologically Realistic Human Motion Using Joint Torque Actuation. *ACM Trans. Graph.* 38, 4, Article 72 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3322966

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)*.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *CoRR* abs/1710.10196 (2017). arXiv:1710.10196 http://arxiv.org/abs/1710.10196

Liyiming Ke, Matt Barnes, Wen Sun, Gilwoo Lee, Sanjiban Choudhury, and Siddhartha S. Srinivasa. 2019. Imitation Learning as f-Divergence Minimization. *CoRR* abs/1905.12888 (2019). arXiv:1905.12888 http://arxiv.org/abs/1905.12888

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. arXiv:http://arxiv.org/abs/1312.6114v10 [stat.ML]

Naveen Kodali, Jacob D. Abernethy, James Hays, and Zsolt Kira. 2017. How to Train Your DRAGAN. *CoRR* abs/1705.07215 (2017). arXiv:1705.07215 http://arxiv.org/abs/1705.07215

Taesoo Kwon and Jessica K. Hodgins. 2017. Momentum-Mapped Inverted Pendulum Models for Controlling Dynamic Human Motions. *ACM Trans. Graph.* 36, 4, Article 145d (Jan. 2017), 14 pages. https://doi.org/10.1145/3072959.2983616

Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. 2002. Interactive Control of Avatars Animated with Human Motion Data. *ACM Trans. Graph.* 21, 3 (July 2002), 491–500. https://doi.org/10.1145/566654.566607

Kyungho Lee, Seyoung Lee, and Jehee Lee. 2018. Interactive Character Animation by Learning Multi-Objective Control. *ACM Trans. Graph.* 37, 6, Article 180 (Dec. 2018), 10 pages. https://doi.org/10.1145/3272127.3275071

Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. 2019. Scalable Muscle-Actuated Human Simulation and Control. *ACM Trans. Graph.* 38, 4, Article 73 (July 2019), 13 pages. https://doi.org/10.1145/3306346.3322972

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010a. Data-Driven Biped Control. *ACM Trans. Graph.* 29, 4, Article 129 (July 2010), 8 pages. https://doi.org/10.1145/1778765.1781155

Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. 2010b. Motion Fields for Interactive Character Locomotion. *ACM Trans. Graph.* 29, 6, Article 138 (Dec. 2010), 8 pages. https://doi.org/10.1145/1882261.1866160

Sergey Levine, Yongjoon Lee, Vladlen Koltun, and Zoran Popović. 2011. Space-Time Planning with Parameterized Locomotion Controllers. *ACM Trans. Graph.* 30, 3, Article 23 (May 2011), 11 pages. https://doi.org/10.1145/1966394.1966402

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.

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. 2020. Character Controllers Using Motion VAEs. *ACM Trans. Graph.* 39, 4, Article 40 (July 2020), 12 pages. https://doi.org/10.1145/3386569.3392422

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, and Baining Guo. 2012. Terrain runner: control, parameterization, composition, and planning for highly dynamic motions. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 154.

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. *ACM Trans. Graph.* 29, 4, Article 128 (July 2010), 10 pages. https://doi.org/10.1145/1778765.1778865

Ying-Sheng Luo, Jonathan Hans Soeseno, Trista Pei-Chun Chen, and Wei-Chao Chen. 2020. CARL: Controllable Agent with Reinforcement Learning for Quadruped Locomotion. *ACM Trans. Graph.* 39, 4, Article 38 (July 2020), 10 pages. https://doi.org/10.1145/3386569.3392433

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. 2020. Learning Latent Plans from Play. In *Proceedings of the Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 100)*, Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (Eds.). PMLR, 1113–1132. http://proceedings.mlr.press/v100/lynch20a.html

X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley. 2017. Least Squares Generative Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 2813–2821. https://doi.org/10.1109/ICCV.2017.304

Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. 2019. Neural Probabilistic Motor Primitives for Humanoid Control. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJl6TjRcY7

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 http://arxiv.org/abs/1707.02201

Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. 2020. Catch and Carry: Reusable Neural Controllers for Vision-Guided Whole-Body Tasks. *ACM Trans. Graph.* 39, 4, Article 39 (July 2020), 14 pages. https://doi.org/10.1145/3386569.3392474

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. 2018. Which Training Methods for GANs do actually Converge?. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm Sweden, 3481–3490. http://proceedings.mlr.press/v80/mescheder18a.html

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of Complex Behaviors through Contact-Invariant Optimization. *ACM Trans. Graph.* 31, 4, Article 43 (July 2012), 8 pages. https://doi.org/10.1145/2185520.2185539

Igor Mordatch, Jack M. Wang, Emanuel Todorov, and Vladlen Koltun. 2013. Animating Human Lower Limbs Using Contact-Invariant Optimization. *ACM Trans. Graph.* 32, 6, Article 203 (Nov. 2013), 8 pages. https://doi.org/10.1145/2508363.2508365

Uldarico Muico, Yongjoon Lee, Jovan Popović, and Zoran Popović. 2009. Contact-Aware Nonlinear Control of Dynamic Characters. In *ACM SIGGRAPH 2009 Papers* (New Orleans, Louisiana) *(SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, Article 81, 9 pages. https://doi.org/10.1145/1576246.1531387

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Haifa, Israel) *(ICML'10)*. Omnipress, Madison, WI, USA, 807–814.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc., 271–279. https://proceedings.neurips.cc/paper/2016/file/cedebb6e872f539bef8c3f919874e9d7-Paper.pdf

Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning Predict-and-Simulate Policies from Unorganized Human Motion Data. *ACM Trans. Graph.* 38, 6, Article 205 (Nov. 2019), 11 pages. https://doi.org/10.1145/3355089.3356501

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018a. DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills. *ACM Trans. Graph.* 37, 4, Article 143 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201311

Xue Bin Peng, Glen Berseth, and Michiel van de Panne. 2016. Terrain-adaptive Locomotion Skills Using Deep Reinforcement Learning. *ACM Trans. Graph.* 35, 4, Article 81 (July 2016), 12 pages. https://doi.org/10.1145/2897824.2925881

Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Trans. Graph.* 36, 4, Article 41 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073602

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. 2019a. MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 3681–3692. http://papers.nips.cc/paper/8626-mcp-learning-composable-hierarchical-control-with-multiplicative-compositional-policies.pdf

Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. 2018b. SFV: Reinforcement Learning of Physical Skills from Videos. *ACM Trans. Graph.* 37, 6, Article 178 (Nov. 2018), 14 pages.

Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. 2019b. Variational Discriminator Bottleneck: Improving Imitation Learning, Inverse RL, and GANs by Constraining Information Flow. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HyxPx3R9tm

Dean A. Pomerleau. 1988. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *Proceedings of the 1st International Conference on Neural Information Processing Systems (NIPS'88)*. MIT Press, Cambridge, MA, USA, 305–313.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR* abs/1511.06434 (2015). arXiv:1511.06434 http://arxiv.org/abs/1511.06434

Marc H. Raibert and Jessica K. Hodgins. 1991. Animation of Dynamic Legged Locomotion. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*. Association for Computing Machinery, New York, NY, USA, 349–358. https://doi.org/10.1145/122718.122755

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning *(Proceedings of Machine Learning Research, Vol. 15)*, Geoffrey Gordon, David Dunson, and Miroslav Dudík (Eds.). JMLR Workshop and Conference Proceedings, Fort Lauderdale, FL, USA, 627–635. http://proceedings.mlr.press/v15/ross11a.html

Alla Safonova and Jessica K. Hodgins. 2007. Construction and Optimal Search of Interpolated Motion Graphs. *ACM Trans. Graph.* 26, 3 (July 2007), 106–es. https://doi.org/10.1145/1276377.1276510

H. Sakoe and S. Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, 1 (1978), 43–49. https://doi.org/10.1109/TASSP.1978.1163055

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved Techniques for Training GANs. *CoRR* abs/1606.03498 (2016). arXiv:1606.03498 http://arxiv.org/abs/1606.03498

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2015. 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 http://arxiv.org/abs/1707.06347

SFU. [n.d.]. SFU Motion Capture Database. http://mocap.cs.sfu.ca/.

Dana Sharon and Michiel van de Panne. 2005. Synthesis of Controllers for Stylized Planar Bipedal Walking. In *Proc. of IEEE International Conference on Robotics and Animation*.

Kwang Won Sok, Manmyung Kim, and Jehee Lee. 2007. Simulating Biped Behaviors from Human Motion Data. *ACM Trans. Graph.* 26, 3 (July 2007), 107–es. https://doi.org/10.1145/1276377.1276511

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural State Machine for Character-Scene Interactions. *ACM Trans. Graph.* 38, 6, Article 209 (Nov. 2019), 14 pages. https://doi.org/10.1145/3355089.3356505

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

Jie Tan, Yuting Gu, C. Karen Liu, and Greg Turk. 2014. Learning Bicycle Stunts. *ACM Trans. Graph.* 33, 4, Article 50 (July 2014), 12 pages. https://doi.org/10.1145/2601097.2601121

Jeff Tang, Howard Leung, Taku Komura, and Hubert Shum. 2008. Emulating human perception of motion similarity. *Computer Animation and Virtual Worlds* 19 (08 2008), 211–221. https://doi.org/10.1002/cav.260

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. 2018. DeepMind Control Suite. *CoRR* abs/1801.00690 (2018). arXiv:1801.00690 http://arxiv.org/abs/1801.00690

Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Generative Adversarial Imitation from Observation. *CoRR* abs/1807.06158 (2018). arXiv:1807.06158 http://arxiv.org/abs/1807.06158

Adrien Treuille, Yongjoon Lee, and Zoran Popović. 2007. Near-Optimal Character Animation with Continuous Control. In *ACM SIGGRAPH 2007 Papers* (San Diego, California) *(SIGGRAPH '07)*. Association for Computing Machinery, New York, NY, USA, 7–es. https://doi.org/10.1145/1275808.1276386

Michiel van de Panne, Ryan Kim, and Eugene Flume. 1994. Virtual Wind-up Toys for Animation. In *Proceedings of Graphics Interface '94*. 208–215.

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. https://doi.org/10.1145/2601097.2601192

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. 2009. Optimizing Walking Controllers. In *ACM SIGGRAPH Asia 2009 Papers* (Yokohama, Japan) *(SIGGRAPH Asia '09)*. Association for Computing Machinery, New York, NY, USA, Article 168, 8 pages. https://doi.org/10.1145/1661412.1618514

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

Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. 2017. Robust Imitation of Diverse Behaviors. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., 5320–5329. https://proceedings.neurips.cc/paper/2017/file/044a23cadb567653eb51d4eb40acaa88-Paper.pdf

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters. *ACM Trans. Graph.* 39, 4, Article 33 (July 2020), 12 pages. https://doi.org/10.1145/3386569.3392381

Yuting Ye and C. Karen Liu. 2010. Synthesis of Responsive Motion Using a Dynamic Model. *Computer Graphics Forum* (2010). https://doi.org/10.1111/j.1467-8659.2009.01625.x

Wenhao Yu, Greg Turk, and C. Karen Liu. 2018. Learning Symmetric and Low-Energy Locomotion. *ACM Trans. Graph.* 37, 4, Article 144 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201397

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-Adaptive Neural Networks for Quadruped Motion Control. *ACM Trans. Graph.* 37, 4, Article 145 (July 2018), 11 pages. https://doi.org/10.1145/3197517.3201366

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3* (Chicago, Illinois) *(AAAI'08)*. AAAI Press, 1433–1438.

Victor Brian Zordan and Jessica K. Hodgins. 2002. Motion Capture-Driven Simulations That Hit and React. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Antonio, Texas) *(SCA '02)*. Association for Computing Machinery, New York, NY, USA, 89–96. https://doi.org/10.1145/545261.545276

APPENDIX

A TASKS

In this section, we provide a detailed description of each task, and the task reward functions used during training.

Target Heading: In this task, the objective for the character is to move along a target heading direction d^* at a target speed v^*. The goal input for the policy is specified as $\mathrm{g}_t = (\tilde{\mathrm{d}}_t^*, v^*)$, with $\tilde{\mathrm{d}}_t^*$ being the target direction in the character's local coordinate frame. The task-reward is calculated according to:

$$r_t^G = \exp\left(-0.25\left(v^* - \mathrm{d}^* \cdot \dot{\mathrm{x}}_t^{\mathrm{com}}\right)^2\right), \tag{11}$$

where $\dot{x}_t^{\mathrm{com}}$ is the center-of-mass velocity of the character at time step t, and the target speed is selected randomly between $v^* \in [1, 5]$m/s. For slower moving styles, such as Zombie and Stealthy, the target speed is fixed at 1m/s.

Target Location: In this task, the character's objective is to move to a target location x^*. The goal $\mathrm{g}_t = \tilde{\mathrm{x}}_t^*$ records the target location in the character's local coordinate frame. The task-reward is given by:

$$\begin{aligned} r_t^G = {} & 0.7 \exp\left(-0.5||\mathrm{x}^* - \mathrm{x}_t^{\mathrm{root}}||^2\right) \\ & + 0.3 \exp\left(-\left(\max\left(0,\ v^* - \mathrm{d}_t^* \cdot \dot{\mathrm{x}}_t^{\mathrm{com}}\right)\right)^2\right). \end{aligned} \tag{12}$$

Here, $v^* = 1m/s$ specifies a minimum target speed at which the character should move towards the target, and the character will not be penalized for moving faster than this threshold. d_t^* is a unit vector on the horizontal plane that points from the character's root to the target.

Dribbling: To evaluate our system on more complex object manipulation tasks, we train policies for a dribbling task, where the objective is for the character to dribble a soccer ball to a target location. The reward function is given by:

$$r_t^G = 0.1 r_t^{\mathrm{cv}} + 0.1 r_t^{\mathrm{cp}} + 0.3 r_t^{\mathrm{bv}} + 0.5 r_t^{\mathrm{bp}} \tag{13}$$

$$r_t^{\mathrm{cv}} = \exp\left(-1.5 \max\left(0,\quad v^* - \mathrm{d}_t^{\mathrm{ball}} \cdot \dot{\mathrm{x}}_t^{\mathrm{com}}\right)^2\right) \tag{14}$$

$$r_t^{\mathrm{cp}} = \exp\left(-0.5\ ||\mathrm{x}_t^{\mathrm{ball}} - \mathrm{x}_t^{\mathrm{com}}||^2\right) \tag{15}$$

$$r_t^{\mathrm{bv}} = \exp\left(-\max\left(0,\quad v^* - \mathrm{d}_t^* \cdot \dot{\mathrm{x}}_t^{\mathrm{ball}}\right)^2\right) \tag{16}$$

$$r_t^{\mathrm{bp}} = \exp\left(-0.5\ ||\mathrm{x}_t^* - \mathrm{x}_t^{\mathrm{com}}||^2\right). \tag{17}$$

r_t^{cv} and r_t^{cp} encourages the character to move towards and stay near the ball, where $\mathrm{x}_t^{\mathrm{ball}}$ and $\dot{\mathrm{x}}_t^{\mathrm{ball}}$ represent the position and velocity of the ball, $\mathrm{d}_t^{\mathrm{ball}}$ is a unit vector pointing from the character to the ball, and $v^* = 1\mathrm{m/s}$ is the target velocity at which the character should move towards the ball. Similarly, r_t^{bv} and r_t^{bp} encourages the character to move the ball to the target location, with d_t^* denoting a unit vector pointing from the ball to the target. The goal $\mathrm{g}_t = \tilde{\mathrm{x}}_t^*$ records the relative position of the target location with respect to the character. The state s_t is augmented with additional features that describe the state of the ball, including the position $\tilde{\mathrm{x}}_t^{\mathrm{ball}}$, orientation $\tilde{\mathrm{q}}_t^{\mathrm{ball}}$, linear velocity $\tilde{\dot{\mathrm{x}}}_t^{\mathrm{ball}}$, and angular velocity $\tilde{\dot{\mathrm{q}}}_t^{\mathrm{ball}}$ of the ball in the character's local coordinate frame.

Strike: Finally, to further demonstrate our approach's ability to compose diverse behaviors, we consider a task where the character's objective is to strike a target using a designated end-effector (e.g. hands). The target may be located at various distances from the character. Therefore, the character must first move close to the target before striking it. These distinct phases of the task entail different optimal behaviors, and thus requires the policy to compose and transition between the appropriate skills. The goal $\mathrm{g}_t = (\tilde{\mathrm{x}}_t^*, h_t)$ records the location of the target $\tilde{\mathrm{x}}_t^*$ in the character's local coordinate frame, along with an indicator variable h_t that specifies if the target has already been hit. The task-reward is partitioned into three phases:

$$r_t^G = \begin{cases} 1, & \text{target has been hit} \\ 0.3\, r_t^{\mathrm{near}} + 0.3, & ||\mathrm{x}^* - \mathrm{x}_t^{\mathrm{root}}|| < 1.375m\ . \\ 0.3\, r_t^{\mathrm{far}}, & \text{otherwise} \end{cases} \tag{18}$$

If the character is far from the target x^*, r_t^{far} encourages the character to move to the target using a similar reward function as the Target Location task (Equation 12). Once the character is within a given distance of the target, r_t^{near} encourages the character to strike the target with a particular end-effector,

$$r_t^{\mathrm{near}} = 0.2 \exp\left(-2||\mathrm{x}^* - \mathrm{x}_t^{\mathrm{eff}}||^2\right) + 0.8\ \mathrm{clip}\left(\frac{2}{3}\mathrm{d}_t^* \cdot \dot{\mathrm{x}}_t^{\mathrm{eff}},\ 0,\ 1\right),$$

where $\mathrm{x}_t^{\mathrm{eff}}$ and $\dot{\mathrm{x}}_t^{\mathrm{eff}}$ denote the position and velocity of the end-effector, and d_t^* is a unit vector pointing from the character's root to the target. After striking the target, the character receives a constant reward of 1 for the remaining time steps.

Obstacles: Finally, we consider tasks that involve visual perception and interaction with more complex environments, where the character's goal is to traverse an obstacle filled environment, while maintaining a target speed. Policies are trained for two types of environments. 1) An environment containing a combination of obstacles including gaps, steps, and overhead obstacles that the character must duck under. 2) An environment containing stepping stones that requires more precise contact planning. Examples of the environment are available in Figure 1 and 3. The task-reward is the same as the one used for the Target Heading task (Equation 11), except the target heading is fixed along the direction of forward progress. In order for the policy to perceive the upcoming obstacles, the state is augmented with a 1D height-field of the upcoming terrain. The height-field records the height of the terrain at 100 sample locations, uniformly spanning 10m ahead of the character.

B AMP HYPERPARAMETERS

Hyperparameter settings used in the AMP experiments are available in Table 4. For single-clip imitation tasks, we found that a smaller discount factor $\gamma = 0.95$ allows the character to more closely imitate a given reference motion. A larger discount factor $\gamma = 0.99$ is used for experiments that include additional task objective, since these tasks may require longer horizon planning, such as *Dribble* and *Strike*.

Table 4. AMP hyperparameters.

Parameter	Value
w^G Task-Reward Weight	0.5
w^S Style-Reward Weight	0.5
w^{gp} Gradient Penalty	10
Samples Per Update Iteration	4096
Batch Size	256
K Discriminator Batch Size	256
π Policy Stepsize (Single-Clip Imitation)	2×10^{-6}
π Policy Stepsize (Tasks)	4×10^{-6}
V Value Stepsize (Single-Clip Imitation)	10^{-4}
V Value Stepsize (Tasks)	2×10^{-5}
D Discriminator Stepsize	10^{-5}
$\mathcal{B}$ Discriminator Replay Buffer Size	10^5
γ Discount (Single-Clip Imitation)	0.95
γ Discount (Tasks)	0.99
SGD Momentum	0.9
GAE(λ)	0.95
TD(λ)	0.95
PPO Clip Threshold	0.2

C LATENT SPACE MODEL

The latent space model follows a similar architecture as Peng et al. [2019a] and Merel et al. [2019]. During pretraining, an encoder $q(\mathrm{z}_t|\mathrm{g}_t)$ maps a goal g_t to a distribution over latent variables z_t. A latent encoding $\mathrm{z}_t \sim q(\mathrm{z}_t|\mathrm{g}_t)$ is then sampled from the encoder distribution and passed to the policy as an input $\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{z}_t)$. The latent distribution is modeled as a Gaussian distribution $q(\mathrm{z}_t|\mathrm{g}_t) = \mathcal{N}(\mu_q(\mathrm{g}_t), \Sigma_q(\mathrm{g}_t))$, with mean $\mu_q(\mathrm{g}_t)$ and diagonal covariance matrix $\Sigma_q(\mathrm{g}_t)$. The encoder is trained jointly with the policy using the following objective:

$$\arg\max_{\pi,q} \quad \mathbb{E}_{p(\tau|\pi,q)}\left[\sum_{t=0}^{T-1}\gamma^t r_t\right] + \lambda \mathbb{E}_{p(\mathrm{g}_t)}\left[\mathrm{D}_{\mathrm{KL}}\left[q(\cdot|\mathrm{g}_t)||p_0\right]\right]. \tag{19}$$

$\tau = \{(\mathrm{s}_t, \mathrm{a}_t, \mathrm{g}_t, r_t)_{t=0}^{T-1}, \mathrm{s}_T, \mathrm{g}_T\}$ represents the goal-augmented trajectory, where the goal g_t may vary at each time step, and

$$p(\tau|\pi, q) = p(\mathrm{g}_0)p(\mathrm{s}_t)\prod_{t=0}^{T-1}\left((\mathrm{g}_{t+1}|\mathrm{g}_t)p(\mathrm{s}_{t+1}|\mathrm{s}_t, \mathrm{a}_t)\right. \tag{20}$$

$$\left.\int_{\mathrm{z}_t}\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{z}_t)q(\mathrm{z}_t|\mathrm{g}_t)d\mathrm{z}_t\right) \tag{21}$$

is the likelihood of a trajectory under a given policy π and encoder q. Similar to a VAE, we include a KL-regularizer with respect to a variational prior $p_0(\mathrm{z}_t) = \mathcal{N}(0, I)$ and coefficient λ. The policy and encoder are trained end-to-end with PPO using the reparameterization trick [Kingma and Welling 2014]. Once trained, the latent space model can be transferred to downstream tasks by using $\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{z}_t)$ as a low-level controller, and then training a separate high-level controller $u(\mathrm{z}_t|\mathrm{s}_t, \mathrm{g}_t)$ that specifies latent encodings z_t for the low-level controller. The parameters of π are fixed, and a new high-level controller u is trained for each downstream task.

During pretrainig, the latent space model is trained using a motion imitation, where the objective is for the character to imitate a corpus of motion clips. A reference motion is selected randomly at the start of each episode, and a new reference motion is selected every 5-10s. The goal $\mathrm{g}_t = (\hat{q}_{t+1}, \hat{q}_{t+2})$ specifies target poses from the reference motion at the next two time steps.

The networks used for π and u follow a similar architecture as the networks used for the policies trained with AMP. The encoder q is modeled by a network consisting of two hidden layers, with 512 and 256 hidden units, followed by a linear output layer for $\mu_q(\mathrm{g}_t)$ and $\Sigma_q(\mathrm{g}_t)$. The size of the latent encoding is set to 16D. Hyperparameter settings are available in Table 5.

Table 5. Latent space model hyperparameters.

Parameter	Value
Latent Encoding Dimension	16
λ KL-Regularizer	10^{-4}
Samples Per Update Iteration	4096
Batch Size	256
π Policy Stepsize (Pre-Training)	2.5×10^{-6}
u Policy Stepsize (Downstream Task)	10^{-4}
V Value Stepsize	10^{-3}
γ Discount (Pre-Training)	0.95
γ Discount (Downstream Task)	0.99
SGD Momentum	0.9
GAE(λ)	0.95
TD(λ)	0.95
PPO Clip Threshold	0.2

D SPATIAL COMPOSITION

Our experiments have so far focused primarily on *temporal* compositions of skills, where a character performs different skills at different points in time in order to fulfill particular task objectives, such as walking to a target and then punching it. In this section, we explore settings that require *spatial* composition of multiple skills, where the task requires a character to perform different skills simultaneously. To evaluate AMP in this setting, we consider a compositional task where a character needs to walk along a target heading direction while also waving its hand at a target height. The motion prior is trained using a dataset consisting of both walking motions and waving motions, but none of the motion clips show examples of walking and waving at the same time. Therefore, the onus is on the policy to spatially compose these different classes of skills in order to fulfill the two disparate objectives simultaneously.

In this task, the character has two objectives: 1) a target heading objective for moving along a target direction d^* at a target speed v^*, 2) and a waving objective for raising its right hand to a target height y^*. The goal input for the policy is given by $\mathrm{g}_t = (\tilde{\mathrm{d}}_t^*, v^*, y^*)$, with $\tilde{\mathrm{d}}_t^*$ being the target direction in the character's local coordinate frame. The composite reward is calculated according to:

$$r_t^G = 0.5r_t^{\text{heading}} + 0.5r_t^{\text{waving}}, \tag{22}$$

where r_t^{heading} the same as the reward used for the Target Heading task equation 11, and r_t^{wave} is specified according to:

$$r_t^{\text{wave}} = \exp\left(-16\left(y_t^{\text{hand}} - y^*\right)^2\right), \tag{23}$$

where y_t^{hand} is the height of character's right hand.

To evaluate AMP's ability to compose disparate skills spatially, we compare policies trained using both walking and waving motions, with policies trained with only walking motions or only waving motions. Table 6 compares the performance of the different policies with respect to the target heading and waving objectives. Although the motion prior was not trained with any reference motions that show both walking and waving at the same time, the policy was able to discover behaviors that combine these different skills, enabling the character to walk along different directions while also waving its hand at various heights. The policies trained with only walking motions tend to ignore the waving objective, and exhibit solely walking behaviors. Policies trained with only the waving motion are able to fulfill the waving objective, but learns a clumsy shuffling gait in order to follow the target heading direction. These results suggest that AMP does exhibit some capability for spatial composition different skills. However, the policies trained with both datasets can still exhibit some unnatural behaviors, particularly when the target height for the hand is high.

Table 6. Performance of policies trained using different dataset on a spatial compositional task that combines following a target heading and waving the character's hand at a target height. The normalized task returns for each objective is averaged across 100 episodes for each model. The model trained with both walking and waving motions achieves relatively high rewards on both objectives, while the models trained with only one type of motions perform well only on one of the objectives.

Dataset (Size)	**Heading Return**	**Waving Return**
Wave (51.7s)	0.683 ± 0.195	0.949 ± 0.144
Walk (229.7s)	0.945 ± 0.192	0.306 ± 0.378
Wave + Walk (281.4s)	0.885 ± 0.184	0.891 ± 0.202

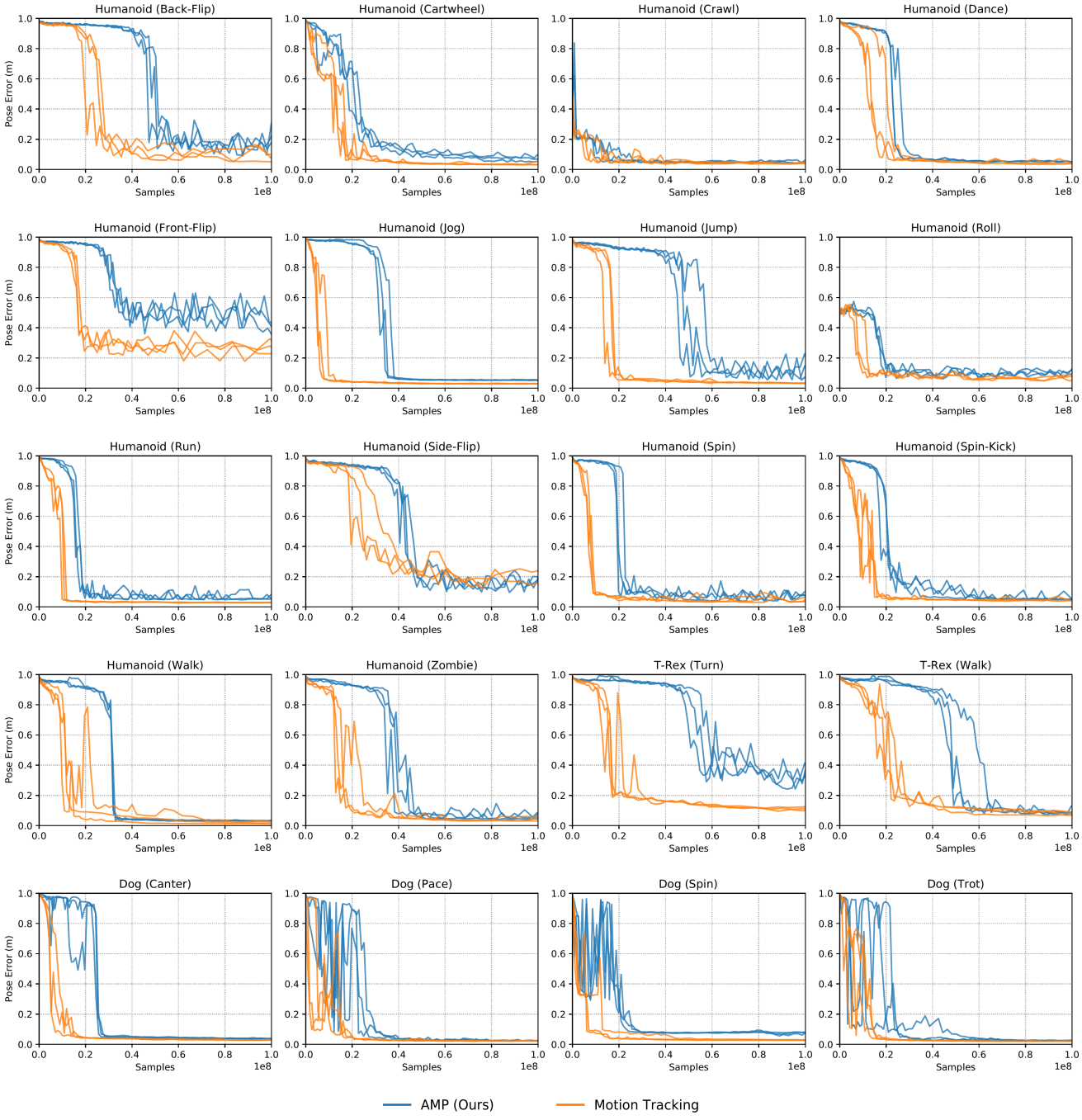


Fig. 9. Learning curves comparing AMP to the motion tracking approach proposed by Peng et al. [2018a] (Motion Tracking) on the single-clip imitation tasks. 3 policies initialized with different random seeds are trained for each method and motion. AMP produces results of comparable quality when compared to prior tracking-based methods, without requiring a manually designed reward function or synchronization between the policy and reference motion.

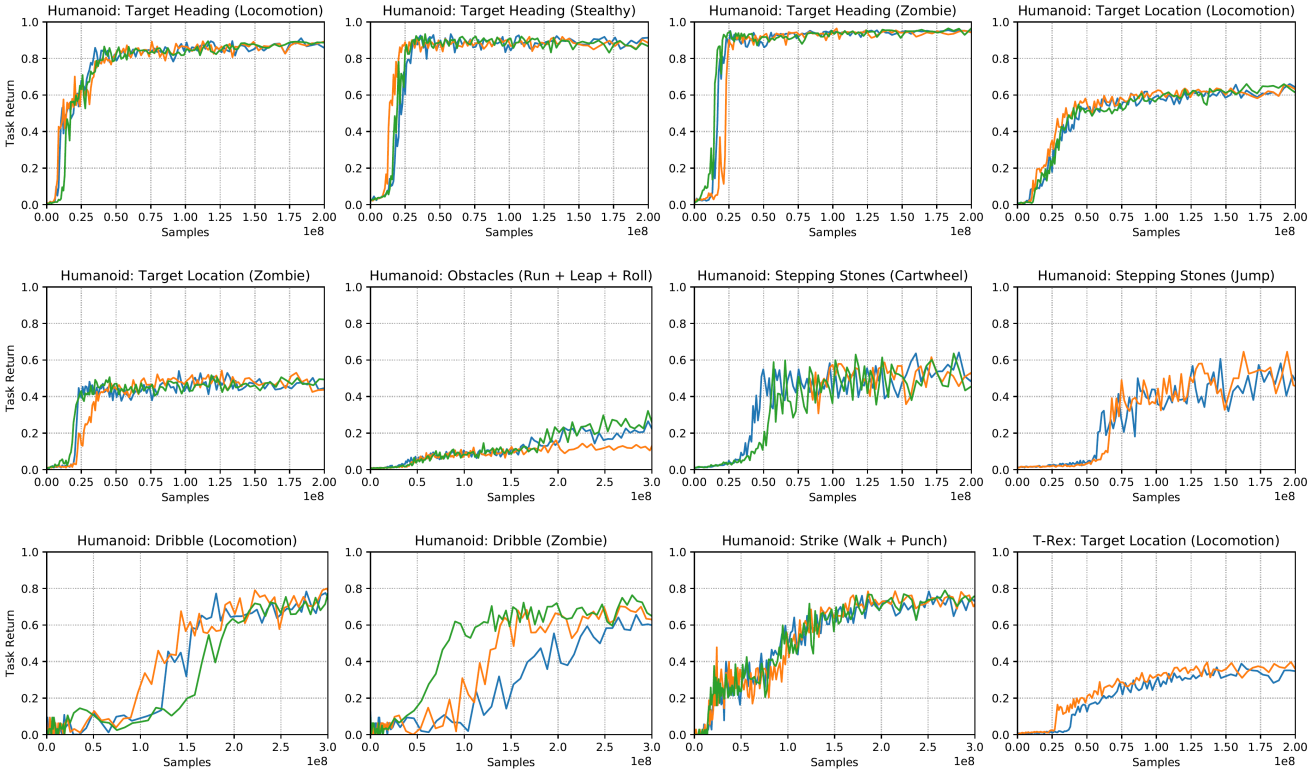


Fig. 10. Learning curves of applying AMP to various tasks and datasets.