Composite Motion Learning with Task Control

PEI XU, Clemson University, USA and Roblox, USA
XIUMIN SHANG, University of California, Merced, USA
VICTOR ZORDAN, Roblox, USA and Clemson University, USA
IOANNIS KARAMOUZAS, Clemson University, USA

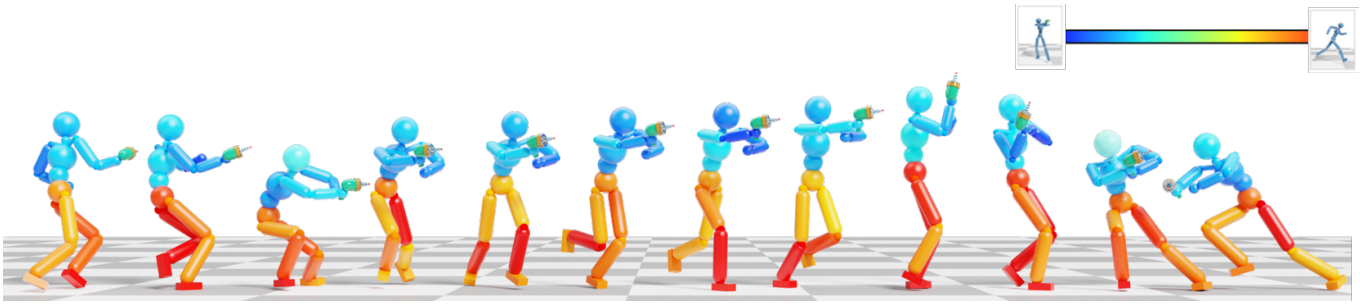


Fig. 1. Example of a physically simulated character performing composite motion with locomotion and aiming a weapon. The colors show the automatic mixing of the combined inputs that change dynamically over time based on the state. As indicated in the inset, red denotes body parts that are vital for locomotion while blue for aiming respectively. Our multi-objective approach learns this mixture along with imitation from two disparate reference motions and two goal-directed task rewards for each action.

We present a deep learning method for composite and task-driven motion control for physically simulated characters. In contrast to existing data-driven approaches using reinforcement learning that imitate full-body motions, we learn decoupled motions for specific body parts from multiple reference motions simultaneously and directly by leveraging the use of multiple discriminators in a GAN-like setup. In this process, there is no need of any manual work to produce composite reference motions for learning. Instead, the control policy explores by itself how the composite motions can be combined automatically. We further account for multiple task-specific rewards and train a single, multi-objective control policy. To this end, we propose a novel framework for multi-objective learning that adaptively balances the learning of disparate motions from multiple sources and multiple goal-directed control objectives. In addition, as composite motions are typically augmentations of simpler behaviors, we introduce a sample-efficient method for training composite control policies in an incremental manner, where we reuse a pre-trained policy as the meta policy and train a cooperative policy that adapts the meta one for new composite tasks. We show the applicability of our approach on a variety of challenging multi-objective tasks involving both composite motion imitation and multiple goal-directed control.

CCS Concepts: • **Computing methodologies** → **Animation**; *Physical simulation*; *Reinforcement learning*.

Authors' addresses: Pei Xu, Clemson University, 1240 Supply Street, North Charleston, SC, 29405, USA, Roblox, USA, peix@clemson.edu; Xiumin Shang, University of California, Merced, USA, xshang@ucmerced.edu; Victor Zordan, Roblox, USA, Clemson University, USA, vbz@clemson.edu; Ioannis Karamouzas, Clemson University, USA, ioannis@clemson.edu.

0730-0301/2023/8-ART $15.00
https://doi.org/10.1145/3592447

Additional Key Words and Phrases: character animation, physics-based control, motion synthesis, reinforcement learning, multi-objective learning, incremental learning, GAN

ACM Reference Format:
Pei Xu, Xiumin Shang, Victor Zordan, and Ioannis Karamouzas. 2023. Composite Motion Learning with Task Control. *ACM Trans. Graph.* 42, 4 (August 2023), 18 pages. https://doi.org/10.1145/3592447

1 INTRODUCTION

Despite significant advancements in physics-based character control, the majority of existing techniques rely on reference data consisting of motion capture recordings of an expert performing the behavior of interest [Bergamin et al. 2019; Chentanez et al. 2018; Lee et al. 2019; Park et al. 2019; Peng et al. 2018, 2022, 2021; Won et al. 2020; Xu and Karamouzas 2021]. While such reference data is paramount to train motor control policies that lead to natural and robust control, in this paper, we are interested in synthesizing composite behaviors for physically simulated humanoids by combining *multiple* motion capture reference clips into the training of a single policy. Further, we augment these imitation controllers with task-specific rewards to train the policy to accomplish specific functional tasks at the same time. To this end, we propose a novel multi-objective learning framework that builds *composite* motion behaviors through multiple discriminators, each with its own distinct reference motion as well as task-level control. Our framework is based on deep reinforcement learning, and allows us to adaptively balance the learning of disparate motions from multiple sources and also multiple goal-directed control objectives.

The motivation for this technique is twofold. First, humans are capable of sophisticated behaviors, including performing multiple tasks simultaneously, such as walking and gesturing or using a

mobile phone. To accomplish this with virtual characters, existing control approaches need to be extended to accommodate the ability to train with multiple objectives as a goal. Second, with limited exception, most current control frameworks rely on imitation with the style of a behavior being derived from reference motion examples. Our aim is to be able to combine examples automatically through what we call "composite motion control" to avoid the need to continuously seek new example motions for every new permutation of combined behaviors. We also explore the ability to add multiple task objectives to support our aim of multi-objective control.

The core difference of our approach from existing imitation learning approaches is decoupling full-body control during training, turning imitation and goal-directed full-body training into a multi-objective learning framework. To this end, we propose a modification to generative adversarial networks (GANs) to accommodate multiple discriminators (for each subtask in the desired end behavior) and to incorporate the mixing of the behaviors as a part of the training. In this way, we sidestep the need to dictate weights for combining the subtasks as well as the need to shape careful reward functions manually for each new composite behavior. In addition, as we expect composite motions to often be augmentations from simpler behaviors, we introduce a method for learning composite motion control policies from existing policies through *incremental learning*. To this end, we train a meta policy, for example for walking, and then train a new policy to *cooperate* with the meta policy, producing a composite motion control policy significantly faster than learning from scratch. Thus, we can quickly add on to walking new activities from reference data such as punching or waiving, even if we do not have examples of these activities being combined previously with the meta policy.

One naive approach to produce the composite motions we target is to blend motion capture clips to produce a single new motion, and perform traditional imitation learning from there. This suggested technique may be plausible for simple composite behaviors, like waiving an arm while walking as the two behaviors do not use the same joints, nor do they influence each other greatly, and therefore the blending can be done by simple splicing in a way that is fixed over time. Even so, there is no guarantee of physical plausibility without subsequent training – and the approach does not scale for more complex behaviors which may have more complicated tradeoffs between body parts used, especially over time. In contrast, our approach offloads the need to create this weighting as it is produced automatically by the policy as a part of the dictated action. Likewise, the output of our system is automatically guaranteed to be physically valid. Finally, our approach also has the capability to add task-directed goals, such as walk to a specified location, which is not possible without significant manual effort being added to the naive approach described.

Overall, this paper makes the following contributions:

- We introduce a novel approach for physics-based character control that decouples full-body control in order to learn imitation and task goals from disparate sources and across distinct body parts.
- To this end, we extend GAN-style reinforcement learning and introduce a multi-objective learning framework to support multiple discriminators and automatic weighting of imitation and goal-driven subtask rewards.
- We propose an incremental learning scheme that uses a meta-policy from an existing behavior to augment the behavior with new subtasks, producing a composite motion control policy that can be learned significantly faster than learning from scratch. Our scheme automatically learns weights across the body that are state dependent in order to effectively mix the original behavior with a new subtask in a temporally dynamic fashion.

2 BACKGROUND AND RELATED WORK

2.1 Physics-Based Character Control

Developing controllers for physically simulated humanoids has wide applications in computer graphics, robotics, and biomechanics. Over the years, a number of trajectory optimization approaches for physics-based control have been proposed that leverage heuristics or feedback rules [Coros et al. 2010; De Lasa and Hertzmann 2009; Wampler et al. 2014; Ye and Liu 2010a; Zordan et al. 2014], including open-loop control schemes[Liu et al. 2015, 2010; Mordatch et al. 2012], close-loop feedback control [da Silva et al. 2017; Mordatch and Todorov 2014] and model predictive control approached [Hämäläinen et al. 2015; Kwon and Hodgins 2010; Tassa et al. 2012, 2014]. Given the difficulty in controller design, which often involves multiple optimization objectives, data-driven methods using demonstrations from real humans has also drawn a lot of attention [Da Silva et al. 2008; Kwon and Hodgins 2017; Lee et al. 2010; Liu et al. 2016, 2012; Muico et al. 2009; Sok et al. 2007; Yin et al. 2007; Zordan and Hodgins 2002].

In recent years, with the advancement of machine learning techniques, deep reinforcement learning frameworks have gained a lot of popularity for training physics-based character controllers. While some works [Karpathy and Van De Panne 2012; Won et al. 2018; Xie et al. 2020; Yu et al. 2018] purely rely on reward functions designed heuristically or using curriculum learning to perform control and encourage the character to act in an expected, human-preferred style, most recent works leverage motion capture data to perform imitation learning in order to generate high-fidelity, life-like motions. DeepLoco [Peng et al. 2017] employs a hierarchical controller to perform walking-style imitation in navigation tasks for a physically simulated character. DeepMimic [Peng et al. 2018] combines imitation learning with goal-conditioned learning, and enables a physics-based character to learn a motor skill from a reference motion collected by motion capture or handcrafted by artists. Chentanez et al. [2018] explore the training of recovery policies that would prevent the character from deviating significantly from the reference motion. While the aforementioned works rely on a phase variable to synchronize with the reference motion, DReCon [Bergamin et al. 2019] utilizes a motion matching technique to find the target pose from a collection of reference motions dynamically in response to user control input.

Besides relying on direct tracking of reference motions, researchers have offered a number of ways to extend the use of reference data in various ways. For example, Park et al. [2019] leverage the kinematic characteristics of unorganized motions to generate target poses for

the control policy to imitate. UniCon [Wang et al. 2020] adopts a similar strategy, where a high-level motion scheduler is employed to provide the target pose for the low-level character controller. MotionVAE [Ling et al. 2020] employs data-driven generative models using variational autoencoders to generate target motion poses for a reinforcement learning based controller. A similar model is employed by Won et al. [2022] and tested with various goal-directed downstream tasks. To ensure synthesis of desired motions, these approaches rely on carefully designed reward functions to assess the controlled character motion. Drawn from GAIL [Ho and Ermon 2016; Merel et al. 2017], AMP [Peng et al. 2021] and ICCGAN [Xu and Karamouzas 2021] avoid manually designing reward functions by exploiting the idea of generative adversarial network (GAN) and relying on a discriminator to obtain the imitation reward for training.

Beyond the simple use of full-body motions, many works explore motion generation by combining together multiple basic motions with respect to different body parts [Alvarado et al. 2022; Jang et al. 2022, 2008; Liu and Hodgins 2018; Soga et al. 2016; Starke et al. 2021; Yazaki et al. 2015]. However, these works focus on the editing and synthesis of motion animation or using inverse kinematic solvers, and do not work well with current frameworks for controlling physically simulated characters using reinforcement learning. To date, existing works for physics-based character control solely focus on the learning of full-body motions. As complementary to such works, in this paper, we target composite motion learning from multiple references without needing to generate any target full-body motion for tasks involving both goal-directed control and imitation control.

2.2 Training Efficiency

Characters employed during physics-based control typically are highly articulated with many degrees of freedom defined in continuous action spaces. Given the vast feasible choices of action, controlling so many degrees of freedom is essentially ambiguous, resulting in control problems that are under specified and highly dimensional. A qualified control policy usually needs millions of samples for training. The time consumption depends on the exploited algorithms and the motion complexity, varying from tens of hours to several days. While some works such as [Yang and Yin 2021] explore approaches to speed up the training by improving the reinforcement learning algorithm itself, a lot of attention has been recently drawn on sample-efficient training by reusing pre-trained policies or action models for fast new motion learning. For example, many recent approaches employ mixture of experts (MoE) models [Peng et al. 2019; Won et al. 2020, 2021], where a batch of pre-trained expert policies are exploited to provide primitive actions that are combined by a newly trained policy to generate the final actions. Other approaches explore using pre-trained latent space models such as variational autoencoders [Ling et al. 2020; Won et al. 2022] and GAN-based models [Peng et al. 2022] to facilitate the training of a control policy. In such approaches, the latent space model encapsulates a variety of reference motions and is used by the control policy to generate motions for a specific task. The works in [Merel et al. 2019, 2020] combine MoE with a latent space model and rely on an encoder-decoder architecture to perform distillation for motion learning. Ranganath et al. [2019] utilize principal component analysis to extract coactivations from reference motions and use them as the atomic actions for motor skill learning.

Despite achieving impressive results, exploring the latent space or learning how to combine expert policies is not always easier compared to performing exploration directly in the original action space. We note that all of these works focus only on reusing models that provide full-body motions. In contrast, we propose an incremental learning approach that allows a newly trained policy to take only partial actions from a pre-trained policy, and add on that to generate composite motions. Our approach can largely reduce the training time for composite and multi-objective tasks involving multiple imitation and goal-directed objectives as compared to training from scratch.

2.3 Multi-Objective Control

In multi-objective character control, the reward function of the underlying optimization problem is expressed as the weighted sum of multiple, possibly competing, goals. Depending on the task in hand, we seek for objective terms that encourage the character to accomplish behavior goals, follow reference motion and/or style, adopt certain behavior characteristics such as low energy movement, attaining specified goals, etc., resulting in an extensive list of objective terms (see [Abe et al. 2007; Macchietto et al. 2009; Muico et al. 2009; Peng et al. 2018; Wu and Zordan 2010; Ye and Liu 2010a,b] for some examples). But how we handle all these competing objectives to create coherent, natural, and coordinated control remains an open question. A common solution is to employ a manual weighting scheme based on intuition, experience, and trial and error. However, such approaches often require excessive, often tricky manual effort to obtain desired results. While prioritized-based schemes have been employed that optimize each term in the reward function based on a given priority [De Lasa and Hertzmann 2009; De Lasa et al. 2010], such schemes cannot automatically address the problem of multiple competing objectives.

This problem becomes worse within a reinforcement learning setting, as small changes in the reward function can have a significant impact on the resulting behavior. It may need laborious work to finetune the weight of each objective to ensure that the control policy can effectively balance the learning of multiple objectives in a desired way. For tasks with hierarchical objectives, hierarchical reinforcement learning with multiple controllers can be employed, where a different controller is selected at different task levels [Clegg et al. 2018; Nachum et al. 2019; Peng et al. 2017; Xie et al. 2020]. However, such approaches cannot work for nonhierarchical tasks, where different objective terms need to simultaneously be optimized such as when the character has to perform composite motion imitation and goal-directed control as in our problem domain. In our approach, we propose the use of a multi-critic optimization scheme, where each objective is regarded as an independent task and is assigned a separate critic. By evaluating each objective independently, the contribution (gradient) of each objective can be normalized into the same scale, and, thus, the control policy will be updated toward each objective at the same pace. As such, we avoid scalarizing and

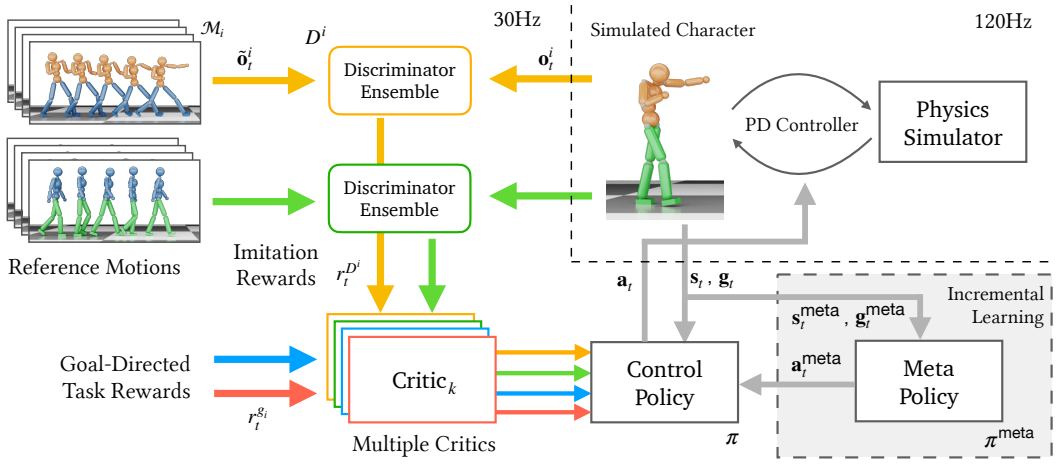


Fig. 2. Overview of the proposed system for composite motion learning with task control. Under the framework of reinforcement learning combined with a GAN-like structure for motion imitation, our approach employs a multi-critic architecture to train a physics-based controller involving multiple objectives. Based on this system, we further propose an optional incremental learning scheme that allows the control policy to fast learn new composite motions and tasks by reusing a pre-trained, meta policy.

weighting the rewards or priorities of multiple objectives. In addition, our approach provides a simple solution to adaptively balance the multiple objectives during policy updating without needing to find or estimate the Pareto front.

3 OVERVIEW

Our approach enables a physically simulated character to perform composite motions through imitating partial-body motions from multiple reference sources directly and simultaneously. This scheme turns the full-body motion imitation task into a multi-objective optimization problem, to which we can further introduce extra objectives for goal-directed control. We refer to Fig. 2 for an overview of our proposed system for composite motion learning with task control. We employ a GAN-like structure combined with reinforcement learning to train the control policy imitating the given reference motions. As such, we do not have to manually design a reward function for imitation learning or explicitly track a target pose from the reference motions. To learn composite motions, we decouple the full-body motion into several partial-body groups each of which imitates its own references. Based on this GAN-like structure, we propose a multi-objective learning framework that exploits multiple critics at the same time to help the control policy learn from multiple objectives, involving both composite motion imitation and goal-directed task control in a balanced way (Section 4). To accelerate training, we further consider an optional incremental learning scheme that reuses a pre-trained policy as the meta policy and allows a cooperative policy to adapt the meta one for new composite tasks (Section 5).

4 COMPOSITE MOTION LEARNING

Given a physically simulated character, we seek to train a control policy $\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g}_t)$ that simultaneously imitates motions from multiple reference ones, each focusing on specific body parts, while possibly completing specific goal tasks. At each time step t, the control policy takes the character state s_t and a dynamic goal state variable g_t as the input and outputs the control signal (action) a_t. We let g_t be an empty variable if no goal-directed control is involved. In the following, we detail our proposed approach for training π that decouples full-body motion allowing imitation performance to be evaluated and improved with respect to specific body parts, and converts the underlying composite motion learning problem into a multi-objective optimization problem.

4.1 Full-Body Motion Decoupling

At each time step t, we represent the character pose as $\mathcal{P}_t := \{(p_l, q_l, \dot{p}_l, \dot{q}_l)|t\}_{l=1}^{N_{\text{link}}}$, where $p_l \in \mathbb{R}^3$ and $q_l \in \mathbb{R}^4$ are the position and orientation (measured in the unit of quaternion) of each body link respectively, and $\dot{p}_l \in \mathbb{R}^3$ and $\dot{q}_l \in \mathbb{R}^3$ are the linear and angular velocities respectively. Given the geometry model and joint constraints of the simulated character, this representation can be converted into a joint space one defined by the skeletal joints' local position and velocity and the root's global position and orientation.

Let $\mathcal{M} \supset \{\tilde{\mathcal{P}}_t\}_t$ be the collection of reference motions which may contain multiple clips of pose trajectories $\{\tilde{\mathcal{P}}_t\}_t$ as the reference. To perform imitation learning, existing approaches either use a carefully designed reward function to compute the error between $\mathcal{P}_{t+1}$ and $\tilde{\mathcal{P}}_{t+1}$ [Bergamin et al. 2019; Chentanez et al. 2018; Park et al. 2019; Peng et al. 2018; Won et al. 2020], or employ an evaluator to assess the transfer $\mathcal{P}_t \rightarrow \mathcal{P}_{t+1}$ without explicitly comparing to any specific poses in the reference motions [Merel et al. 2017; Peng et al. 2021; Xu and Karamouzas 2021]. The former approaches usually need a motion tracking or generation mechanism to retrieve $\tilde{\mathcal{P}}_{t+1}$ from the reference motions. The latter typically build on the framework of adversarial generative networks (GANs) and rely on a discriminator to evaluate the transfer. Some approaches take poses from more than one frame during imitation performance evaluation in order to apply more constraints on the pose trajectory.

Nevertheless, all these approaches leverage the full-body character pose $\mathcal{P}_t$ and reference pose $\tilde{\mathcal{P}}_t \in \mathcal{M}$ to perform imitation learning, and thus intend to learn the full-body motions in $\mathcal{M}$.

To learn composite motions, ideally, we want the simulated character's partial body motions to come from different reference sources at a given time step t, i.e., the transfer of pose trajectory $\mathcal{P}^i_{t-n_i:t} \rightarrow \mathcal{P}^i_{t+1}$ should satisfy

$$\{\mathcal{P}^i_{t-n_i}, \cdots, \mathcal{P}^i_t, \mathcal{P}^i_{t+1}\} \subset \mathcal{M}^i, \tag{1}$$

where $\mathcal{P}^i_t \subset \mathcal{P}_t$ is a partial-body pose from the simulated character, and $\mathcal{M}^i \supset \{\tilde{\mathcal{P}}^i_t\}_t$ is the reference motion collection containing only poses of the partial body group i. The full-body motion is constrained by using multiple $\mathcal{M}^i$ at the same time. Here, we follow Xu and Karamouzas [2021] and use a pose trajectory having $n_i + 2$ frames for imitation performance evaluation. The larger n_i is, the stricter the evaluation will be, as an error occurring at an earlier time step would negatively influence the evaluation of the following steps.

Typical partial body groups for a humanoid character would be the upper and lower body, arms, and torso. For example, we can let $\mathcal{M}^{\text{upper}}$ be a collection of greeting motions involving the upper body (arms, hands, torso and head), and $\mathcal{M}^{\text{lower}}$ be walking motions involving the lower body (pelvis, legs and feet). Then, the full body motion is expected to be the composite of $\mathcal{M}^{\text{upper}}$ (greeting) and $\mathcal{M}^{\text{lower}}$ (walking). To coordinate the motions from multiple body groups, we can let $\mathcal{P}^i_t$ and some other partial-body poses $\mathcal{P}^j_t$ share some common body link states. For example, let $\mathcal{P}^{\text{upper}}_t$ and $\mathcal{P}^{\text{lower}}_t$ share the state of one leg to avoid ipsilateral walking. Correspondingly, the leg state should be included in both $\mathcal{M}^{\text{upper}}$ and $\mathcal{M}^{\text{lower}}$ for the control policy to learn. We refer to Sections 6 and 7 for body splitting schemes used in our experiments, including typical upper and lower body decoupling schemes and more tailored ones for specific tasks such as juggling while walking. After decoupling the character's full-body motion into multiple sets of $\{\mathcal{P}^i_t\}_t$, we perform imitation learning with respect to each body group independently, where the control policy is expected to explore how to combine partial-body motions by itself without needing any full-body, composite motions to be provided as the reference.

4.2 Imitation Learning

To perform imitation learning, we build our approach off of GAN-like frameworks [Ho and Ermon 2016; Merel et al. 2017], which utilize a discriminator to evaluate imitation performance and generate reward signals for policy optimization using reinforcement learning algorithms. However, instead of using only one discriminator to perform full-body imitation performance evaluation, we employ multiple discriminators simultaneously, each of which deals with a body part group i associated with a collection of partial-body reference motions $\mathcal{M}^i$. Based on this framework, we can avoid designing reward functions to compute the imitation error for each specific body part group. Furthermore, each discriminator can take only its own interested body link states as input during training. Therefore, the provided $\mathcal{M}^i$ can still be a collection of full-body motions, but there is no need to explicitly generate any partial-body motions during preprocessing.

To stabilize the adversarial training process, we introduce a hinge loss [Lim and Ye 2017], gradient penalty term [Gulrajani et al. 2017], and an ensemble technique for training of discriminators as proposed in [Xu and Karamouzas 2021]. Following the literature, given o^i_t as the observation sampled from the simulated character and $\tilde{\mathrm{o}}^i_t$ as that sampled from the reference motions $\mathcal{M}^i$, the i-th ensemble of N discriminators, $D^i = \{D^i_n | n = 1, \cdots, N\}$ is trained using the loss function:

$$\mathcal{L}_{D^i} = \frac{1}{N}\sum_{n=1}^{N}\Big(\mathrm{E}_t\left[\max(0, 1 + D^i_n(\mathrm{o}^i_t))\right] + \mathrm{E}_t\left[\max(0, 1 - D^i_n(\tilde{\mathrm{o}}^i_t))\right] + \lambda^{\text{GP}}\mathrm{E}_t\left[(||\nabla_{\hat{\mathrm{o}}^i_t} D^i_n(\hat{\mathrm{o}}^i_t)||_2 - 1)^2\right]\Big) \tag{2}$$

where $\hat{\mathrm{o}}^i_t = \alpha \mathrm{o}^i_t + (1-\alpha)\tilde{\mathrm{o}}^i_t$ with $\alpha \sim \textsc{Uniform}(0, 1)$ and λ^{GP} is gradient penalty coefficient.

According to Eq. 1, we define the observation space of a discriminator as

$$\mathrm{o}^i_t := \{\mathcal{P}^i_{t-n_i}, \cdots, \mathcal{P}^i_t, \mathcal{P}^i_{t+1}\}. \tag{3}$$

In principle, the discriminator relies on o^i_t to evaluate the control policy's performance during the state-action-state transition $(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_{t+1})$. The observation space theoretically should satisfy $\mathrm{o}^i_t \subseteq \{\mathrm{s}_t, \mathrm{s}_{t+1}\}$. Otherwise, the discriminator may rely on features unknown to the control policy, and thus it cannot effectively evaluate the policy's performance. Given that the control policy π in our formulation is still a full-body control policy, we simply define s_t as a full-body motion state:

$$\mathrm{s}_t := \{\mathcal{P}_{t-n}, \cdots, \mathcal{P}_t\} \tag{4}$$

where $n \geq n_i$ for all i. We refer to the Appendix in the supplementary material for more details about the state and observation representation.

The hinge loss function provides a linear evaluation between $[-1, 1]$ to measure the similarity of a given pose trajectory sample o^i_t to any sample in the reference motions. Therefore, we define the reward term that evaluates the policy's imitation performance with respect to $\mathcal{M}^i$ for the body part group i at time t as:

$$r^{D^i}_t(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_{t+1}) = \frac{1}{N}\sum_{n=1}^{N}\textsc{Clip}\left(D^i_n(\mathrm{o}^i_t), -1, 1\right). \tag{5}$$

It must be noted that even though o^i_t and $\tilde{\mathrm{o}}^i_t$ in Eq. 2 have the same subscript t, they are paired only for the gradient penalty computation (last term in Eq. 2). The discriminator ensemble here only evaluates the pose trajectory o^i_t independently, rather than comparing it against any specific target trajectory. Therefore, $\tilde{\mathrm{o}}^i_t$ can be randomly sampled from the reference motions by interpolation.

Overall, by employing multiple discriminator ensembles at each time step t, we will have a set of rewards, $\{r^{D^i}_t\}_{D^i}$, to evaluate the policy's performance of controlling the character to perform composite motions, i.e. simultaneously imitating different sets of reference motions corresponding to specific partial body parts. By doing so, we convert the task of composite motion learning to a multi-objective optimization problem under the framework of reinforcement learning.

4.3 Multi-Objective Learning

We consider policy optimization of a typical on-policy policy gradient algorithm by maximizing

$$\mathcal{L}_\pi = \mathrm{E}_t[A_t \log \pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g}_t)], \tag{6}$$

where s_t and g_t are the given character's and goals' state variables respectively, and A_t is the advantage which is typically estimated by $\{r_\tau\}_{\tau \geq t}$. In the common actor-critic architecture, a separate network (critic) is updated in tandem with the policy network (actor). The critic is employed to provide state-dependent value estimation, $V(\mathrm{s}_t) = \mathrm{E}_\pi[\sum_{\tau \geq t} \gamma^{\tau-t} r_\tau] = \mathrm{E}_\pi[r_t + \gamma V(\mathrm{s}_{t+1})]$, based on which A_t can be estimated with less variance, where γ is the discount factor regulating the importance of the contribution from future steps. To stabilize the training, standardization is often applied on A_t where the standardized advantage $\bar{A}_t$ is used in place of A_t for policy updating.

A typical solution for multi-objective tasks in reinforcement learning is to simply add together all objective-related reward terms, r_t^k, with some weights ω_k, i.e., $r_t = \sum_{k=1}^K \omega_k r_t^k$ for a K-objective problem. In such a way, we still have a scalar reward that can be used with Eq. 6 for policy updating. In practice, though, given that conflicts may exist among the different reward terms, manually tuning the values of ω_k to balance the composite objective of the character is not an intuitive task. For example, we may need the policy to put more effort into learning a difficult partial-body motion, instead of even with a trade-off in learning other motions, rather than only focusing on the easy ones to keep achieving a higher associated reward. In addition, our proposed approach performs reward estimation by employing multiple discriminators simultaneously, which are modeled by neural networks. This scheme brings a lot of uncertainty, as the reward distributions from different discriminators may differ a lot depending on the given reference motions, which could be unpredictable before training. Such a problem would deteriorate if we further introduce a set of goal-directed tasks, each having its own associated reward term which may compete against the imitation reward terms.

To balance the contributions of multiple objectives during policy updating, we propose to model the multi-objective learning problem as a multi-task one, where each objective is taken into account as an *independent task* and has a fixed importance during policy updating. To do so, instead of using $r_t = \sum_k \omega_i r_t^k$, we compute the advantage of A_t^k with respect to $\{r_\tau^k\}_{\tau \geq t}$ independently. Then, the optimization process becomes maximizing

$$\mathcal{L}_\pi = \sum_{k=1}^K \mathrm{E}_t\left[\omega_k \bar{A}_t^k \log \pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g}_t)\right], \tag{7}$$

where $\sum_k \omega_k = 1$ and $\bar{A}_t^k$ is the standardization of A_t^k, i.e.

$$\bar{A}_t^k = \frac{A_t^k - \mathrm{E}_t[A_t^k]}{\sqrt{\mathrm{Var}_t[A_t^k]}}. \tag{8}$$

This optimization process is equal to updating the policy with respect to each objective independently but always at the same scale proportional to ω_k. The introduction of ω_k gives us more flexibility to adjust the contributions toward each objective when conflicts occur during policy updating. However, under our testing, a simple choice of $\omega_k = 1/K$, which means each objective is equally important, works well for most cases. We refer to the Appendix in the supplementary material for the choice of ω_k in our tested composite tasks.

During implementation, we can rewrite Eq. 7 as

$$\mathcal{L}_\pi = \mathrm{E}_t\left[\left(\sum_k \omega_k \bar{A}_t^k\right) \log \pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g}_t)\right] \tag{9}$$

such that the policy update can be done through backward propagation in one pass. From this equation, we can see that the nature of our approach is to introduce a dynamic coefficient constrained by the standard deviation of $\{A_t^k\}_t$ for each objective k. As such, the policy will be updated with respect to each objective adaptively. This separation of objectives leads to a single-policy multi-critic architecture. In Fig. 2, for example, we have two imitation related reward terms (yellow and green) for upper and lower body imitation respectively, and two goal-directed task reward terms (red and blue). Accordingly, we employ four critics denoted by $\textsc{Critic}_k$ in the figure. Each $\textsc{Critic}_k$ only participates in the estimation of A_t^k, and takes the reward associated with the objective k, i.e. $\{r_t^k\}_t$, for training.

Though the policy update is balanced through the proposed multi-critic architecture, the state values, which are decided by $\{r_t^k\}_t$, could differ still drastically with respect to each objective depending on the difficulty of given reference motions or the reward distributions of the goal-related tasks. To mitigate this issue and stabilize the training of critics, we introduce the value normalization scheme of *PopArt* [van Hasselt et al. 2016]. The value target under this scheme is normalized by the moving average and standard deviation for the critic network training. The output of a critic is unnormalized before joining the process of advantage estimation. Besides maintaining a normalizer for value targets, *PopArt* is designed to preserve the output precisely. Namely, with *PopArt*, the output of a critic is identical before and after the normalizer updates given the same input state s_t and g_t. Such a design is to prevent the normalization from affecting the value state estimation, thereby stabilizing the policy training. In our implementation, each critic $\textsc{Critic}_k(\mathrm{s}_t, \mathrm{g}_t)$ has its own normalizer with a scalar scale and shift estimated independently with respect to its associated objective k. As we show in Section 6.6, the introduction of *PopArt* helps improve the policy performance as also demonstrated by previous works [van Hasselt et al. 2016; Yu et al. 2021].

5 INCREMENTAL LEARNING

Besides being able to perform a range of composite motions, humans typically learn such motions in an incremental manner. For example, if we know how to walk, we should be able to quickly learn how to hold our phone while walking. There is no need to relearn walking from scratch. Based on this intuition, we propose an incremental learning scheme for fast composite motion learning. Instead of training a policy completely from scratch, we reuse a pre-trained policy as a meta policy π^{meta} that allows the simulated character to perform a basic set of motions (walking in the previous example). Given π^{meta}, we train a new policy π to cooperate

3. Why is this valid?

This step relies on **linearity of expectation**.

By definition:

$$\sum_k \omega_k \, \mathbb{E}_t[\bar{A}_k^t \log \pi] \;=\; \mathbb{E}_t\left[\sum_k \omega_k \, \bar{A}_k^t \log \pi\right]$$

Since $\log \pi(a_t|s_t, g_t)$ does **not depend on** k (it's the same policy term regardless of which objective we're looking at), we can factor it outside of the sum over k:

$$= \mathbb{E}_t\Big[\Big(\sum_k \omega_k \bar{A}_k^t\Big) \log \pi(a_t|s_t, g_t)\Big]$$

And that's exactly Eq. 9.

1. Why is $\log \pi(a_t|s_t, g_t)$ independent of k?

- The policy π is a **single network** with parameters θ.
- At each state s_t and goal g_t, it outputs a distribution over actions.
- This distribution **doesn't change** depending on which objective k we're evaluating — the policy is the same.

So:

- $\pi(a_t|s_t, g_t)$ is a function of the policy parameters and the (state, goal) pair.
- k only indexes which *objective's advantage estimate* we're pairing with that log-probability.
- Therefore, for a given action at timestep t, the log-probability is the same no matter which objective's advantage we multiply it with.

That's why it can be factored out of the sum over k.

⚠️ Subtlety: You're right to ask, *"But what about randomness across multiple runs/timesteps?"*

- Yes, π is stochastic — different runs may yield different a_t.
- But **once a particular action a_t has been sampled at step t**, its log-probability is fixed.
- The randomness is already "realized"; what we're left with is weighting that log-probability by different advantages.

---

2. Which term is used in backpropagation?

In Eq. 9:

$$L_\pi = \mathbb{E}_t\left[\Big(\sum_k \omega_k \bar{A}_k^t\Big) \log \pi(a_t|s_t, g_t)\right]$$

- **The trainable parameters**: the weights of the **policy network** (θ).
- Backprop computes gradients of L_π w.r.t. θ.
- Concretely:
  - We compute $\nabla_\theta \log \pi(a_t|s_t, g_t)$.
  - Multiply it by the combined weighted advantage $\sum_k \omega_k \bar{A}_k^t$.
  - Average over timesteps.
- This is the **policy gradient theorem** in action.

So what's being trained here = the policy's parameters (the actor network) that determine action probabilities.

---

3. Why Eq. 9 is computationally more effective than Eq. 8

- **Eq. 8 (separate updates):**

$$\sum_k \omega_k \, \mathbb{E}_t\left[\bar{A}_k^t \log \pi(a_t|s_t, g_t)\right]$$

This would mean: for each objective k, compute an expectation, weight it, and backprop **individually**. That implies **K backward passes** per batch.

- **Eq. 9 (combined update):**

$$\mathbb{E}_t\left[\Big(\sum_k \omega_k \bar{A}_k^t\Big) \log \pi(a_t|s_t, g_t)\right]$$

Here, you first combine the weighted advantages into one scalar coefficient per timestep:

$$A_t^{\text{combined}} = \sum_k \omega_k \bar{A}_k^t$$

Then do **one backward pass** using that combined advantage.

So computationally:

- Eq. 8 → K × backprop passes per batch.
- Eq. 9 → 1 × backprop pass per batch.

The trick is purely about regrouping terms under the expectation, using linearity, so the math is equivalent but the compute is reduced.

---

1. What I meant by "once sampled at step t, its probability is fixed"

- The policy $\pi_\theta(a|s, g)$ is a probability distribution over actions.
- At time step t:
  - State = s_t, Goal = g_t.
  - The policy outputs probabilities (like a dice):
    - Move left leg: 0.6
    - Move right leg: 0.3
    - Jump: 0.1
  - Then, we **sample** one action a_t (say "move left leg").

👉 Once we actually sample it, we know **which action was taken** and what its probability was at that moment.
That's what I mean by "realized" → the randomness of sampling has been resolved, so for this update we treat $\log \pi(a_t|s_t, g_t)$ as a fixed number tied to that sampled action.

On another rollout, the sampled action could be different, but **within this step**, it's fixed.

---

2. What is θ, and where is it?

- θ are the **parameters of the policy network** (the weights and biases of the neural net).
- The log-probability term is:

$$\log \pi_\theta(a_t|s_t, g_t)$$

- Here the subscript θ reminds us that the probabilities come from the network with parameters θ.
- During training, backpropagation computes:

$$\nabla_\theta \log \pi_\theta(a_t|s_t, g_t)$$

  which tells us how to nudge the weights θ so that the policy assigns more probability to good actions (positive advantage) and less to bad ones (negative advantage).

So **yes**, the parameters θ are hidden inside the log term, because the log is applied to the probability that the neural network outputs.

---

3. How does backpropagation look here?

For Eq. 9:

$$L_\pi = \mathbb{E}_t\left[\Big(\sum_k \omega_k \bar{A}_k^t\Big) \log \pi_\theta(a_t|s_t, g_t)\right]$$

The gradient is:

$$\nabla_\theta L_\pi = \mathbb{E}_t\left[\Big(\sum_k \omega_k \bar{A}_k^t\Big) \nabla_\theta \log \pi_\theta(a_t|s_t, g_t)\right]$$

- The **coefficient** $\sum_k \omega_k \bar{A}_k^t$ is just a scalar multiplier (no parameters).
- The **thing backprop is applied to** is $\log \pi_\theta$.
- This gradient flows into the neural network that defines π_θ, updating its weights θ.

---

3. Rewards vs Value vs Parameters

- **Rewards** r_t:
  - Come directly from the environment.
  - In this paper, environment reward = outputs from discriminators (imitation) + task rewards.
  - These are **not trainable**, they're given.
- **Value function** $V_\phi(s)$:
  - Approximates the *expected* discounted sum of future rewards.
  - Implemented as a neural network with parameters ϕ.
  - **Learnable** by regression:

$$\min_\phi \Big(V_\phi(s_t) - (r_t + \gamma V_\phi(s_{t+1}))\Big)^2$$

- **Policy** $\pi_\theta(a|s, g)$:
  - Chooses actions, trained via policy gradient.
  - **Learnable** parameters θ.

So there are two networks:

- **Actor** (policy) with parameters θ.
- **Critic** (value function) with parameters ϕ.

ALGORITHM 1: Multi-Objective Incremental Learning

Prepare the meta policy π^{meta};
initialize the policy network π;
initialize the critic network Critic_k where $k = 1, \cdots, K$ given K objectives in the task;
initialize policy replay buffer $\mathcal{T}$ and reward buffer $\mathcal{R}$;
prepare reference motions $\mathcal{M}^i$ for each discriminator ensemble D^i;
while *training does not converge* **do**
  $\mathcal{T} \leftarrow \emptyset, \mathcal{R} \leftarrow \emptyset$;
  for *each environment step t* **do**
    $\mathrm{a}_t^{\text{meta}} \sim \pi^{\text{meta}}(\cdot|\mathrm{s}_t^{\text{meta}}, \mathrm{g}_t^{\text{meta}})$;
    $\mathrm{a}_t \sim \pi(\cdot, |\mathrm{s}_t, \mathrm{g}_t, \mathrm{a}_t^{\text{meta}})$;
    $\mathrm{s}_{t+1}, \mathrm{g}_{t+1}, \mathrm{r}_t^{\mathrm{g}t} \leftarrow$ environment updates with character control signal of a_t;
    extract observation o_t^i from the state pair of s_t and s_{t+1} for each discriminator ensemble D^i;
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{(\mathrm{s}_t, \mathrm{a}_t, \{\mathrm{o}_t^i\}_i)\}$;
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{r_{t+1}^k\}$ for each term k in $\mathrm{r}_t^{\mathrm{g}t}$;
    $\mathrm{s}_t \leftarrow \mathrm{s}_{t+1}$; $\mathrm{g}_t \leftarrow \mathrm{g}_{t+1}$;
    extract $\mathrm{s}_t^{\text{meta}}$ and $\mathrm{g}_t^{\text{meta}}$ from s_t and g_t respectively
  end
  for *each discriminator ensemble* D^i **do**
    draw samples $\tilde{\mathrm{o}}_t^{D^i}$ from $\mathcal{M}^i$;
    update D^i using o_t^i from $\mathcal{T}$ and $\tilde{\mathrm{o}}_t^i$ based on Eq. 2;
    for *each* o_t^i *in* $\mathcal{T}$ **do**
      compute step-wise imitation reward $r_t^{D^i}$ based on Eq. 5;
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{r_t^{D^i}\}$
    end
  end
  for *each reward term collection* $\{r_t^k\}_t$ *in* $\mathcal{R}$ **do**
    compute advantage A_t^k using $\{r_\tau^k\}_{\tau \geq t}$ and state value estimation from $\text{Critic}_k(\mathrm{s}_\tau, \mathrm{g}_\tau)$ unnormalized by *PopArt*;
    compute value target V_t^k based on A_t^k;
    update the normalizer for Critic_k based on V_t^k using *PopArt*;
    get normalized value target $\bar{V}_t^k$ by *PopArt*;
    get normalized advantage $\bar{A}_t^k$ based on Eq. 8
  end
  for *each policy update step* **do**
    update π using $\{(\mathrm{s}_t, \mathrm{a}_t, \{\bar{A}_t^k\}_k)\}_t$ based on Eq. 9;
    update each critic network Critic_k using $\{\bar{V}_t^k\}_t$
  end
end

with the meta policy, performing new composite motions by action addition (holding a phone + walking).

Formally, let $\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g}_t) := \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ denote a Gaussian-based policy. By introducing a meta policy π^{meta}, we define the policy, which is trained to cooperate with π^{meta} for new composite motions as

$$\begin{aligned}\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g}_t, \mathrm{a}_t^{\text{meta}}) &:= \mathcal{N}\left(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2\right) + \mathrm{w}_t \textsc{Stop}\left(\mathrm{a}_t^{\text{meta}}\right) \\ &= \mathcal{N}\left(\boldsymbol{\mu}_t + \mathrm{w}_t \textsc{Stop}\left(\mathrm{a}_t^{\text{meta}}\right), \boldsymbol{\sigma}_t^2\right),\end{aligned} \tag{10}$$

where the weight vector w_t has the same dimension with $\mathrm{a}_t^{\text{meta}}$, and $\mathrm{a}_t^{\text{meta}} \sim \pi^{\text{meta}}(\cdot|\mathrm{s}_t^{\text{meta}}, \mathrm{g}_t^{\text{meta}})$ is drawn from the meta policy. w_t are defined as a set of weights each of which is associated with a DoF in the action space of the meta policy. In our implementation, w_t, $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$ are obtained by a neural network taking s_t and g_t as input, and thus are learnable. We put a "gradient stop" operator, $\textsc{Stop}(\cdot)$, on $\mathrm{a}_t^{\text{meta}}$, which means that the meta policy is fixed and will not be updated with π.

Using this incremental learning scheme, the new, cooperative policy adds its own action to the meta action $\mathrm{a}_t^{\text{meta}}$. The weight vector w_t decides the reliance of π on the meta policy π^{meta} with respect to each DoF in the action space. The bigger an element in w_t is, the more the cooperative policy relies on the meta policy to control the corresponding DoF. As such, π is trained incrementally to learn new composite motions by reusing the meta policy partially. This scheme does not require that $\mathrm{a}_t^{\text{meta}}$ and a_t must have exactly the same dimension, as we can assume zero values for the missing dimensions in $\mathrm{a}_t^{\text{meta}}$ or ignore the extra, uninteresting dimensions in $\mathrm{a}_t^{\text{meta}}$. Compared to a mixture-of-experts (MoE) model, where the action is obtained by a linear combination of the actions from multiple expert policies, our approach focuses on reusing partial-body motions from the meta policy. It would be very difficult for a MoE model to keep, for example, only the lower-body motion of one expert and replace the upper-body motion with that of another expert through a linear combination of the experts' full-body motions.

With the introduction of π^{meta}, we can replace $\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g}_t)$ in Eq. 7 with $\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g}_t, \mathrm{a}_t^{\text{meta}})$, and perform composite motion learning with goal-directed control under our proposed multi-objective learning framework. We refer to Algorithm 1 for the outline of the proposed multi-objective learning framework with incremental learning. To train a composite policy completely from scratch without using incremental learning, we can simply ignore π^{meta} and use $\pi(\mathrm{a}_t|\mathrm{s}_t, \mathrm{g}_t)$ solely in Algorithm 1.

6 EXPERIMENTS

In this section, we experimentally evaluate our approach on multiple challenging composite motion learning tasks. We show that our approach can effectively let motor control policies learn composite motions from multiple reference motions directly without manually generating any full-body motion as reference. Besides evaluating the imitation performance, we also apply our approach on several goal-directed control tasks combined with composite motion learning from unstructured reference data. The results demonstrate that our proposed approach can successfully tackle complex tasks balancing the learning of multiple objectives involving both partial-body motion imitation and goal-directed control. Finally, we perform ablation studies on our proposed multi-objective learning framework and incremental learning scheme.

6.1 Implementation Details

We run physics-based simulations using IsaacGym [Makoviychuk et al. 2021], which supports simulation with a large number of instances simultaneously by leveraging GPU. The simulated humanoid character has 15 body links and 28 DoFs, where the hands are fixed with the forearms and are uncontrollable. In the tasks involving a

chat history: https://chatgpt.com/share/68cc739c-ff80-8009-b83d-cc610405e883

---

1. The three "V's" floating around

1. **Critic's prediction**

$$V_\phi^k(s_t, g_t)$$

   - This is the neural net output (with parameters ϕ).
   - An **estimate**, not ground truth.

2. **Advantage**

$$A_t^k = r_t^k + \gamma V_\phi^k(s_{t+1}, g_{t+1}) - V_\phi^k(s_t, g_t)$$

   - Uses the critic's current predictions at t and $t+1$.
   - Tells the actor: "was this action better/worse than expected?"

3. **Value target** (a supervised training label for the critic)

$$V_t^k = r_t^k + \gamma V_\phi^k(s_{t+1}, g_{t+1})$$

   - This is the *bootstrapped return*.
   - It says: "given what actually happened at step t, the value at this state *should* be this number."
   - It is **not predicted**; it is **constructed** using observed reward + next-step prediction.

↓

---

2. Where the confusion happens

You noticed:

$$V_t^k = A_t^k + V_\phi^k(s_t, g_t)$$

If we substitute in the definition of A_t^k:

$$V_t^k = \left(r_t^k + \gamma V_\phi^k(s_{t+1}, g_{t+1}) - V_\phi^k(s_t, g_t)\right) + V_\phi^k(s_t, g_t)$$

The middle terms cancel:

$$V_t^k = r_t^k + \gamma V_\phi^k(s_{t+1}, g_{t+1})$$

✅ Exactly! This is the **TD target** we train the critic on.

---

3. Why introduce "value target" if it's the same expression?

Because of roles:

- $V_\phi^k(s_t, g_t)$ = critic's current guess (used as baseline, subtracted in advantage).
- V_t^k = supervised "label" we want the critic to match.

The critic update is:

$$\min_\phi \left(V_\phi^k(s_t, g_t) - V_t^k\right)^2$$

So conceptually:

- V_ϕ^k = prediction.
- V_t^k = target.
- Advantage A_t^k = target − prediction.

They look algebraically interchangeable, but in training loop each has a distinct role.

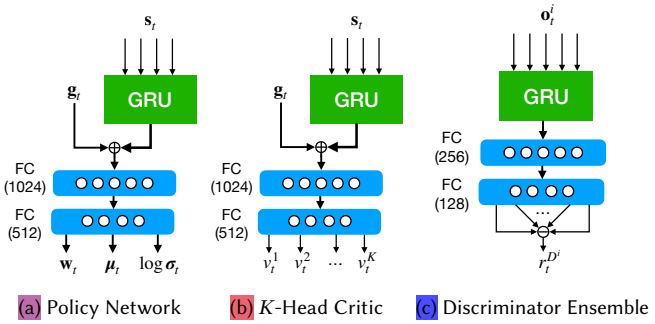


Fig. 3. Network structures. ⊕ denotes the concatenation operator and ⊖ denotes the average operator.

tennis player, we add 3 DoFs on the right wrist joint such that the character can control the racket more agilely, though the racket is fixed on the right hand. The simulation runs at 120Hz and the control policy at 30Hz. Differing from the previous works that employ a stable PD controller [Tan et al. 2011] for character control [Lee et al. 2022, 2021; Park et al. 2019; Peng et al. 2018, 2021; Won et al. 2020, 2022; Xu and Karamouzas 2021] we employ a normal, linear PD servo for faster simulation.

We use PPO [Schulman et al. 2017] as the base reinforcement learning algorithm for policy training and Adam optimizer [Kingma and Ba 2014] to perform policy optimization. To embed the character state s_t and the discriminator observation o_t^i sequentially, we employ a gated recurrent unit (GRU) [Chung et al. 2014] with a 256-dimension hidden state to process these temporal inputs. The embedded character state feature is concatenated with the dynamic goal state g_t if goal-directed control is involved, and then passed through a multilayer perceptron with two full-connected (FC) layers. The control policy is constructed as Gaussian distributions with independent components. The output of the policy network includes the mean $\boldsymbol{\mu}_t$ and standard deviation $\boldsymbol{\sigma}_t$ parameters of the policy distribution as well as a weight vector w_t when incremental learning is exploited. The multiple critics in our multi-objective learning framework are modeled by a multi-head neural network. Similarly to the critic networks, we model a discriminator ensemble using a multi-head network. The outputs are averaged by Eq. 5 to produce the reward signal. All the network structures are shown in Fig. 3, in which we assume that there are K objectives in total. We refer to the Appendix in the supplementary material for the representation of g_t in our designed goal-directed tasks, and all hyperparameters used for policy training.

All the tested policies were trained on a machine equipped with an Nvidia V100 GPU. It typically takes about 1.5h to train a policy using a fixed budget of 20M samples (environment steps), for a pure composite motion imitation task. For complex tasks involving goal-directed control, it takes about 15 to 30 hours and requires about 2×10^8 to 4×10^8 samples to train a policy from scratch. By exploiting our incremental learning scheme to reuse a pre-trained meta policy, we can shorten the training time to about 30 minutes to 2 hours depending on the difficulty of the tasks.

6.2 Data Acquisition

All the motion data used for training are obtained from the LAFAN1 dataset [Harvey et al. 2020] and other commercial and publicly available motion capture datasets recorded at 30Hz. For single-clip imitation, we synthesize short reference motion clips of 1-3 seconds long (cf. Table 1). For tasks with goal-directed control, we extract several collections of motions (cf. Table 2), each of which contains multiple clips of reference motions with lengths varying from about 15 to 70 seconds. The juggling motion involves a single trial of a subject performing juggling while standing on a skate, while the collection of tennis swing motions contains four trials of forehand swings captured from different subjects. We retarget the local joint position from those motion data to our character model without extra manual reprocessing. We demonstrate that policies trained with our approach can perform motion synthesis from unstructured data for goal-directed control, and can explore how to perform composite motions by combining the partial-body motions from the reference motions without needing any manual processing for motion blending.

6.3 Imitation Performance

In Fig. 4, we highlight motion pose snapshots captured from some of our trained policies for composite motion learning. Each composite motion is learned based on two reference motion clips, one for the upper body and the other one for the lower body. From top to bottom, the names of corresponding motions are listed in Table 1. Overall, policies trained with our approach can perform very challenging composite motor skills by using the character's upper and lower body part groups at the same time. For example, in the motion combination of chest open and jumping jack (1st row), the control policy must keep the character's body balanced to perform the chest-open motion during jumping in the air, which is a pretty challenging task even for humans. Similar challenges arise when doing squats with the chest open (3rd row) and lunges with waist twisting (4th row). Besides simply following the two partial-body reference motions at the same time, the control policies must master how the partial motions could be combined such that the full-body motion is physically plausible. In the 4th row, for example, it is impossible for the character to keep twisting its waist while doing lunges at quite different frequencies. Similarly, in the motion combination of punch and walk (6th row) and that of punch and run (7th row), the character's foot has to contact the ground first in order to perform the punch action with the torso leaning forward. The control policy, thereby, must know when the punch action is doable and arrange the motion combination by itself, rather than strictly following the reference motions. Our approach does not require the given reference motions to be perfectly synchronized. The control policies take the character state as input and perform composite motions accordingly. Furthermore, the proposed dynamic sampling rate (see Appendix) allows the control policy to adjust the motion speed within an acceptable range for better motion combining.

To quantitatively evaluate the imitation performance, following previous literature [Harada et al. 2004; Peng et al. 2021; Tang et al. 2008; Xu and Karamouzas 2021], we leverage the technique of fast dynamic time warping (DTW) and measure the imitation error as

Fig. 4. Composite motions learned from multiple single-clip reference motions. The two snapshots shown on the left side of each row are the reference motions for the upper and lower body respectively.

Table 1. Imitation performance when learning composite motions from single clips of reference motions.

Composite Motion	Length [s]	Imitation Error [m]
Chest Open	2.10	0.11 ± 0.02
Front Jumping Jack (lower)	1.80	0.16 ± 0.03
Front Jumping Jack (upper)	1.80	0.30 ± 0.03
Walk In-place	2.10	0.29 ± 0.02
Chest Open	2.10	0.10 ± 0.01
Squat	1.67	0.09 ± 0.01
Waist Twist	3.37	0.15 ± 0.04
Leg Lunge	3.67	0.13 ± 0.02
Hand Waving	1.80	0.06 ± 0.03
Walk	1.10	0.09 ± 0.02
Punch	1.30	0.11 ± 0.02
Walk	1.10	0.10 ± 0.01
Punch	1.30	0.17 ± 0.03
Run	0.76	0.14 ± 0.01

follows:

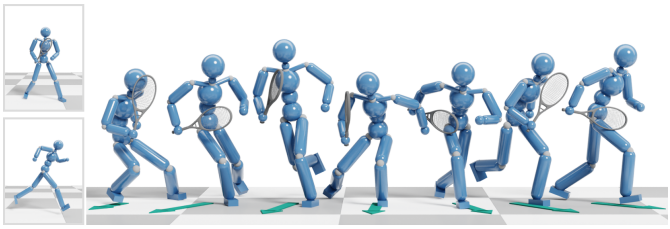
$$e_t = \frac{1}{N^i_{\text{link}}} \sum_{l=1}^{N^i_{\text{link}}} ||p_l - \tilde{p}_l||, \tag{11}$$

where $N^i_{\text{link}} = |\{\mathcal{P}^i_t\}|$ is the number of interesting body links in the i-th body part group, $p_l \in \mathbb{R}^3$ is the position of the body link l in the world space at the time step t, and $\tilde{p}_l$ is the body link's position in the reference motion. The evaluation results are shown in Table 1. Our approach can imitate the reference motions closely and balance the imitation of the two partial-body motions well. As can be seen, there is no big gap between the two imitation errors in a given composite motion combination, which means that policies trained with our approach do not just follow only one reference motion and ignore the other one. In contrast, without using our proposed multi-objective learning framework, the policy could prefer to track only one reference motion that is easy to follow. We refer to Section 6.6 for the related ablation study.

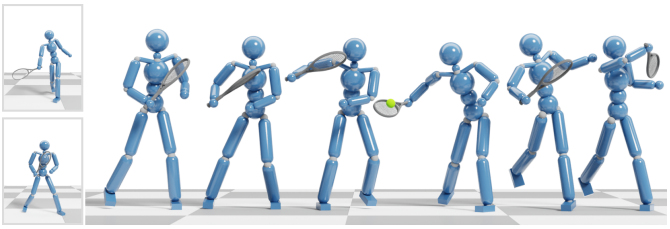
6.4 Goal-Directed Motion Synthesis

To test our approach with more complex tasks involving both composite motion learning and goal-directed control, we designed five goal-directed tasks, as shown in Figs. 5 and 6. In the *Target Heading* and *Target Location* tasks illustrated in Figs. 5a and 5b, the character is asked to respectively go along a target heading direction and toward a target location at a preferred speed. Besides the goal-directed objective, two motion imitation objectives are employed: one is for the lower-body and the other one is for the upper body. Differing from the examples shown in Fig. 4 where the walking and running motions are just single, short clips containing only one gait cycle, here we use a collection of unstructured walking and running motions as the reference for the lower body, as listed in Table 2. In the three examples shown in Fig. 5a, the upper body motions are learned from single reference motion clips, which are chest open, jumping jack, and punch respectively, as depicted by the small snapshots in the figure. In the examples shown in Fig. 5b, we use the motion collection of tennis footwork as the reference

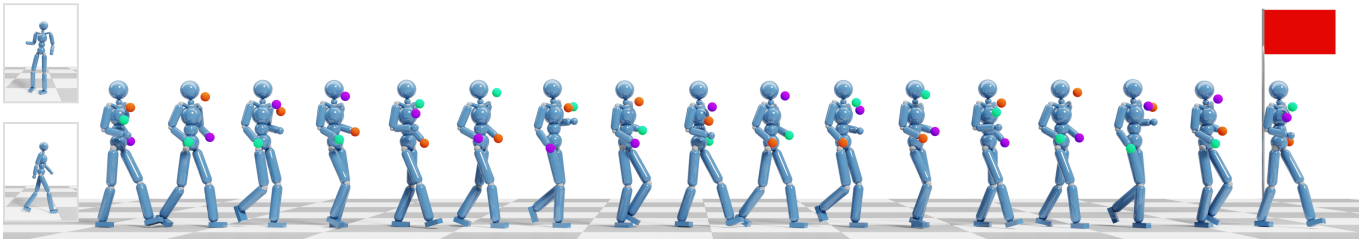
(a) Tasks: Target Heading (Directional Walking with Various Upper-Body Motions)



(b) Task: Target Location (Run) with Tennis Racket Holding



(c) Task: Tennis Swing (Forehand Swing with Footwork)



(d) Task: Target Location (Walk) while Juggling

Fig. 5. Motion synthesis with composite motion learning and goal-directed control. Pose snapshots shown in the small windows are captured from the reference motions.

Table 2. Motion collections used for goal-directed control.

Motion Collection	# of Clips	Length [s]
Crouch	4	88.87
Walk	8	334.07
Run	4	282.87
Tennis Footwork	2	31.67
Tennis Swing	4	13.33
Aiming	2	48.77
Juggling	1	24.63

for the control policy to learn how to hold the racket. This task is relatively harder, as the reference motions for both the upper and lower body are unstructured. While following the reference motions closely, the control policies trained with our approach can effectively coordinate the character's upper and lower body poses to perform the composite motions during goal-steering navigation.

In the task of *Tennis Swing*, the character is expected to hit the ball successfully with a forehand. The provided collection of tennis swing motions contains four trials, where the subject performs forehand swings while standing still. The tennis ball in our implementation is generated randomly in a small region near the character. As such, the control policy has to rely on the lower-body footwork motions to properly adjust the pose and position of the character relative to the tennis ball, while it relies on the upper body swing motions to swing effectively and on time. We note that the goal-directed reward in our design only evaluates the effectiveness of hitting based on the ball's outgoing speed and destination. The motion otherwise is decided completely by the control policy, which leverages two discriminator ensembles to perform imitation learning for the upper and lower body respectively.

The *Tennis Swing* task is challenging, as it is easy for the controlled character to solely hit the ball, but instead it is asked to do so by combining the motions from the reference collection (tennis swing for the upper body and tennis footwork for the lower body). The policy needs some exploration before finding a way to utilize poses from the reference motions to perform swings. In this process, imitation learning would fail if the policy simply tries to pursue a higher reward by simply hitting the ball. However, when the policy is trained using our proposed multi-objective learning framework, it can balance the imitation and goal-directed objectives, and perform forehand swings in the style of the reference motions. Additionally, while we provide only a small set of upper and lower body motions as the reference (cf. Table 2), the control policy successfully learns

(a) Meta Policy Tasks: Target Location (Crouch, Walk and Run)

(b) Incremental Learning Tasks: Directional Aiming while Location Targeting (Crouch, Walk and Run).

Fig. 6. Demonstration of incremental learning tasks, where goal-directed aiming motions are added to various locomotion behaviors from the meta policies.

how to combine the motions automatically to finish the task. In contrast, if we just leverage full-body reference motions, extra work is needed to generate various motions for the policy to learn. In addition, there are not enough demonstrations for the policy to perform tennis swings correctly in a human-like style by utilizing, for example, only standing swing motions without footwork.

Figure 5d shows another challenging composite task: *Target Location* while *Juggling*, where the character needs to juggle three balls while walking to the target location. This composite task involves four objectives: two imitation objectives and two goal-directed tasks of juggling and locomotion. In our experiment, when a ball is relatively close to a hand, it is assumed to be caught by and attached to that hand. The ball is automatically detached from hand at a fixed interval of 20 frames. In order to perform juggling successfully and successively, after a hand releases its ball, it must catch in time a flying target ball which was thrown by the other hand. This task is very challenging, as the control policy must explore how to perform ball throwing and catching in concert with the location-targeting task. Besides the difficulty of throwing and catching balls, the juggling reference motion involves a subject balancing on a skateboard with the body swaying from side to side [1]. This increases the difficulty of composite motion learning to generate normal walking poses. Differing from the other examples that use a lower and upper-body split, here we decouple the body parts into two groups, where one group consists of the character's arms to imitate the juggling motion and the other group includes the rest of the body parts (torso, head, pelvis, and legs) taking the collection of walking motions as reference data. In such a way, our approach can effectively eliminate the body swings in the juggling reference motion, and generate composite motions with the upper body moving naturally during goal-steering navigation.

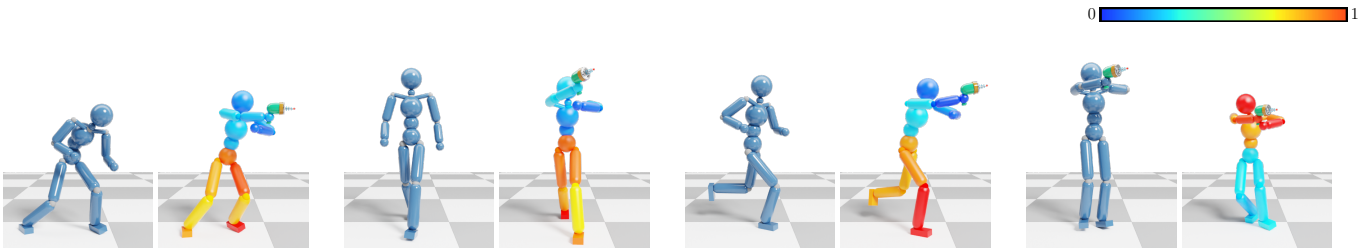
The other goal-directed task explored in this study is *Aiming*, in which the character holds a toy weapon in its right hand and is expected to aim it toward a specific direction. In our experiments, that task is designed mainly to demonstrate the effectiveness of our proposed incremental learning scheme, which will be elaborated in the next section. We refer to the Appendix for the details of the setup of all of our goal-directed tasks, and the supplementary video for related animation results.

6.5 Incremental Learning

In Fig. 6, we show tasks used to test our proposed incremental learning scheme. The first row depicts three meta policies of locomotion, which are trained for the *Target Location* task completely from scratch using our proposed multi-objective learning framework. In contrast to previous examples, there is only one imitation objective about the full-body during training here, as shown by the snapshots on the top-left corner of the figure. In the 2nd row of the figure, we show the cooperative policies that are trained by incremental learning, while reusing the pre-trained, meta policies. In addition to the *Target Location* task, a new goal-directed task of *Aiming* is introduced during training the cooperative policies. The controlled character in this task needs to adjust its right forearm and let the toy pistol aim toward a goal direction specified dynamically.

The goal of this experiment is to demonstrate that the cooperative policies can properly exploit the meta policies to perform styled locomotion behaviors while quickly learning upper-body motions from the newly provided aiming reference motions, which also involve a new goal-directed task that is never seen by the meta policies. In Fig. 7, we visualize the weight vector w_t (cf. Eq. 10) for each DoF

[1] FreeMoCap Project: https://github.com/freemocap/freemocap



(a) Aiming+Crouch (b) Aiming+Walk (c) Aiming+Run (d) Crouch+AimingWalk

Fig. 7. Visualization of the incremental learning weight w_t (cf. Eq.10). The azure character shows the behavior from the meta policy. The colored character is controlled by the cooperative policy. The body link color identifies the weight for the associated DoF. The redder color represents higher weights, which means that the cooperative policy relies more on the meta policy to control the corresponding body parts of the character. The bluer color represents lower weights, which means that the cooperative policy mainly relies on itself to control the related body parts.

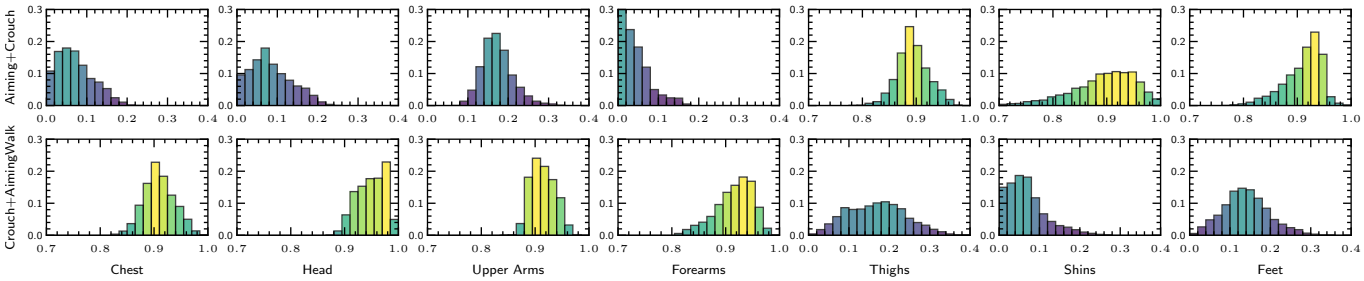


Fig. 8. Distributions of the incremental learning weights w_t for the tasks of Aiming+Crouch and Crouch+AimingWalk (cf. Fig. 7). The x-axis depicts the learned weights and the y-axis shows the corresponding distribution density, normalized by the total number of samples per body part grouping. The color saturation binds the weight range for higher distribution density, with brighter colors highlighting weights greater than 0.5. In the first task, the lower body is mainly controlled by the meta Crouch policy (high weights), while in the second task the AimingWalk meta policy mainly influences the upper body.

by coloring the associated body link. The first three examples show the results obtained when we add the aiming motions to the meta policies of locomotion. The fourth example shows the corresponding result of adding the crouch motion to the meta policy of aiming and walking. As opposed to the previous meta policies, this meta policy has four objectives: two imitation objectives for the upper (aiming) and lower (walking) body respectively, one *Target Location* task and one *Aiming* task.

As shown in the figure, in the three Aiming+Locomotion tasks where the meta policies are pre-trained for locomotion, the cooperative policies rely more on the meta policy for lower-body actions and control the upper-body parts for aiming primarily by themselves. In contrast, in Crouch+AimingWalk, we want the cooperative policy to replace the walking motions from the meta policy with crouching while keeping the upper-body motion of aiming. Here, as can be seen in the fourth case of the figure, the cooperative policy exploits the meta policy to perform aiming actions but performs crouching mainly on its own. In Fig. 8, we also plot the distribution of weights based on the collection of 5,000 consecutive frames from the Aiming+Crouch and Crouch+AimingWalk tasks. The statistical results are consistent with the above studied cases.

As an additional experiment, in Fig. 9, we show that control policies trained with our approach can support the interactive control scheme proposed by Xu and Karamouzas [2021]. In this experiment, we let the character perform a variety of locomotion styles by switching the three trained Aiming+Locomotion policies interactively in response to external control signal provided by the user, and navigate to and aim at the target directions specified by the user dynamically.

6.6 Ablation Studies

We refer to the previous literature of ICCGAN [Xu and Karamouzas 2021] for ablation studies with respect to each component in the employed GAN-like structure for motion imitation, and to [Peng et al. 2021; Xu and Karamouzas 2021] for related analyses on the robustness of control policies trained using GAN-like structures combined with reinforcement learning. Here, we focus on the studies of the proposed multi-objective learning framework and incremental learning scheme.

In Fig. 10, we compare the performance of our proposed multi-objective (MO) learning framework to two baselines using three composite motion learning tasks from Section 4.1. The first baseline leverages our MO learning framework but does not make use of *PopArt* to normalize the value targets of each critic (w/o PopArt). The second baseline simply adds the rewards from the two discriminators together and models the composite motion learning task as a typical reinforcement learning problem (w/o MO). Both baselines are trained with our motion decoupling scheme described in Section 4.1 and simultaneously leverage two discriminators, one for the upper-body motion and one for the lower body. As can be seen from

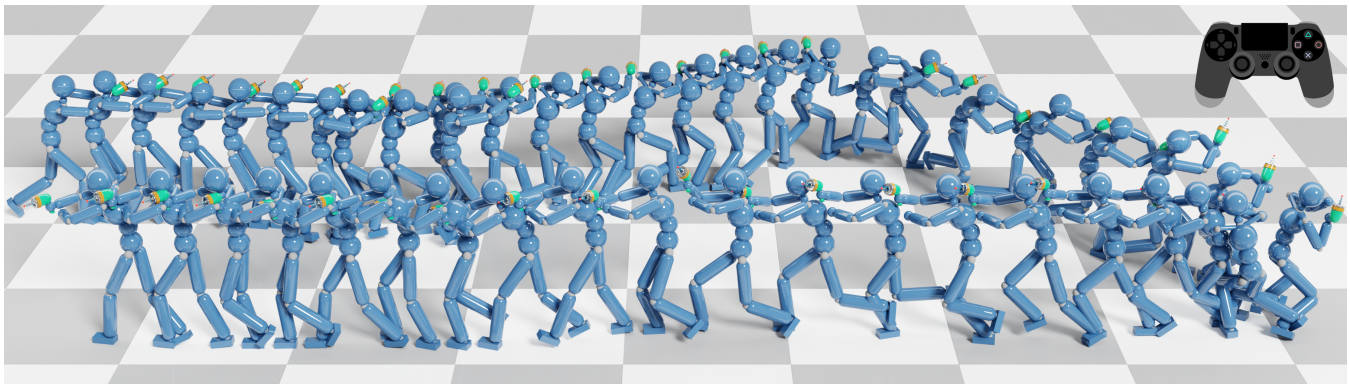


Fig. 9. Interactive control of switching between walking, crouching and running for location targeting while aiming.

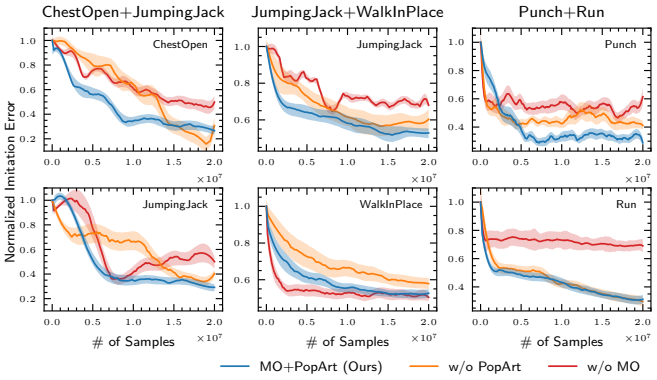


Fig. 10. Learning performance on tasks of composite motion learning from two single-clip reference motions, which are illustrated in Fig. 4. "MO" stands for the proposed multi-objective learning framework detailed in Section 4.3. Colored regions denote mean values ± a standard deviation based on 10 trials.

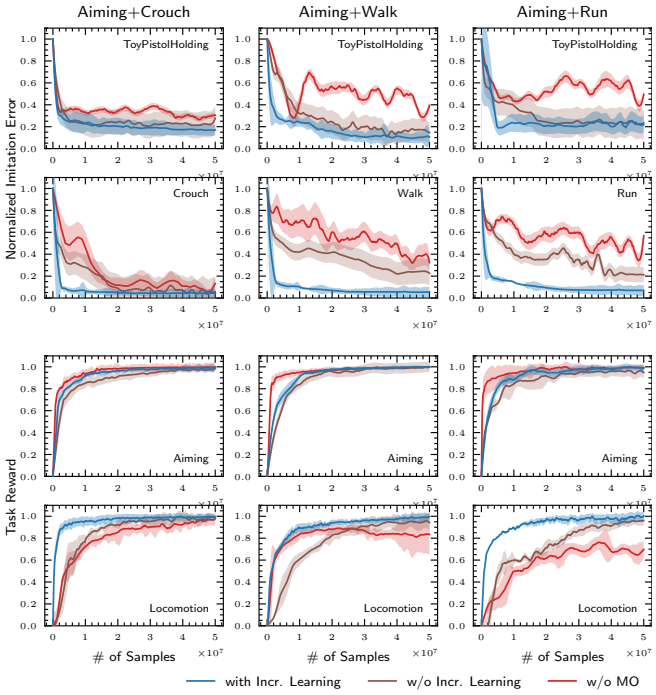


Fig. 11. Learning performance on three composite tasks where each task combines learning from two partial motions while accomplishing two goal objectives. Multi-objective learning in an incremental manner leads to sample-efficient training allowing for high-fidelity composite motion synthesis with goal-directed control. Colored regions denote mean values ± one standard deviation based on 10 trials.

the figure, it is hard for "w/o MO" to balance the learning of the two reference motions. For example, in the ChestOpen+JumpingJack task, as the upper-body (ChestOpen) imitation error goes down, the lower-body (JumpingJack) error increases; in the Punch+Run task, the policy almost gives up on learning how to run, focusing on punching without too much success. In contrast, when leveraging our MO framework either with or without *PopArt*, the imitation errors of the upper and lower body show similar and stable trends, keep decreasing as the training goes on. Additionally, the introduction of *PopArt* typically facilitates better training, allowing for faster convergence speed, lower imitation error, and more robust training achieving similar performance across different trials.

Figure 11 shows the performance of our MO approach with and without exploiting the proposed incremental learning scheme. We also provide comparisons with the "w/o MO" baseline. The tested tasks have four objectives, as described in Section 6.5: two imitation objectives for the upper and lower body respectively, one *Target Location* task for the locomotion and one *Aiming* task. In the cases using incremental learning, we employed a pre-trained, locomotion policy as the meta one. Consistent with the previous ablation study, we can see that the "w/o MO" baseline struggles to balance the different objective terms. Here, the character quickly achieves a high reward for the goal-directed *Aiming* task (3rd row) but fails to complete other objectives, and in particular to account for the motion style provided by the imitation reward terms. For example, the controlled character holds the toy pistol in an unnatural

way compared to the demonstrations in the provided reference motions as indicated by the high imitation error (1st row). While such issues are successfully resolved by our proposed MO framework, learning in a non-incremental way leads to sample inefficient training as compared to learning by leveraging a meta policy. Besides slow speed of convergence, non-incremental training can be time consuming for challenging multi-objective tasks. For example, in the Aiming+Run task, while the case with incremental learning only needs 1.5 hours to finish the training by using about 20 million samples, the non-incremental cases need about 20 more hours for training and will consume about 300 million more samples to achieve a similar performance.

7 LIMITATIONS AND FUTURE WORK

We present a technique for training composite-motion controllers using a multi-objective learning framework that is capable of combining multiple reference examples and task goals to control a physically-simulated character. We demonstrate that our approach can generalize to a large number of examples based on the availability of reference data. Likewise, we show its ability to accomplish simultaneous goal-driven tasks such as aiming at specific targets and moving to a target location with different locomotion styles. Furthermore, we can interactively control such character's actions, pushing the boundary of what is capable for physics-based characters to date.

Of course, there is still more to explore in this space. Our system is currently not well-equipped to handle behaviors which include multiple phases, as the imitation is not phase-locked in any fashion and our discriminators do not distinguish between different stages of an activity. Exploring the potential to add a state machine with state transitions could aid in this capacity [Starke et al. 2019]. Another shortcoming of the approach presently is that we do not account for variation across the humans that recorded the motion clips. This implies that we are introducing bias in the imitation process that may degrade the final quality of the animation. As is, the system is able to make adjustments automatically as needed based on the physical characteristics of the behavior but it cannot distinguish errors that are more stylistic.

In its current form, our system can not create new composite activities without performing additional training. A possible direction for future work is aimed at sidestepping this limitation to directly combine preexisting policies and greatly improve the scalability of trained controllers. That is, to train two (or more) policies independently and combine them at runtime to create a composite motion. Finally, in human motion, composite behaviors go beyond an anticipated split, e.g. the lower and upper body, which is one of the modest underlying assumptions in our current implementation. Instead, humans may enlist body parts and release them fluidly. For example, a well-trained martial artist changes the use of appendages quickly in fighting sequences. We wish to explore this direction in future investigations and believe that our proposed multi-objective learning framework can provide the foundation for such future endeavors.

Although we employed an upper and lower body split in most of our experiments, there is nothing tied to this body decoupling scheme except that it is a practical general choice for deploying the limbs of the whole body. Currently, as long as the subtasks are compatible, our system is capable of combining motions along other body splits. For instance, in the *Juggling+TargetLocation* example discussed in Section 6.4, the trained policy controls the arms for juggling and the rest of the body for walking. Our approach may fail if, for example, the lower limbs are separated due to the requirements of physical balance. As an example, in Fig. 12, we show a failure case where the body is bisected into a left/right split and asked to imitate walking and jumping motions respectively. Such a composite motion is not well-defined, even for humans. We can see that though not falling down, the simulated character cannot imitate the two motions accurately, and instead performs an in-between motion where the character neither jumps up nor walks in an expected fashion.

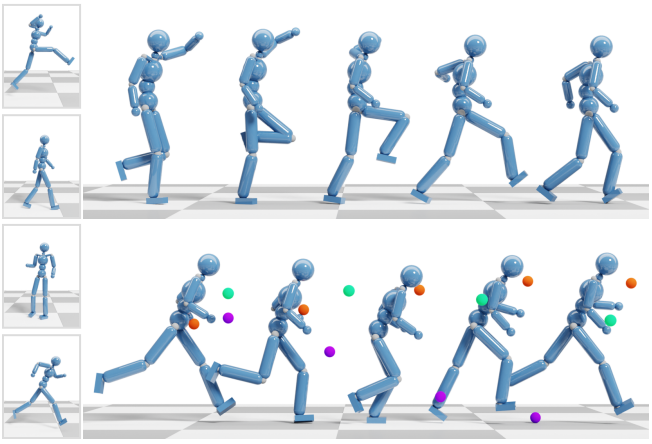


Fig. 12. Failure case study. Top: The character's body is bisected into a left and right group, imitating walking and jumping respectively. Bottom: Juggle while running.

In Fig. 12, we also show another failure case where running reference motions with an average speed of around $3.5m/s$ are provided for the *Juggling+TargetLocotion* task. With the difficulty of juggling while moving at this higher speed, this example is significantly more challenging than the one shown in Fig. 5d. Even though we are able to synthesize the composite motions, the simulated character cannot juggle the balls successfully under these conditions. Currently, our approach cannot identify if a composite motion is compatible on its own, and instead, it relies on a human to combine behaviors with some domain knowledge about the affinity of the mixing and the feasibility of associated goal-directed tasks. Automating this would be a great direction for future work.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant No. IIS-2047632 and by Roblox. We would like to thank Rokoko [2] for providing mocap data for this project.

[2]https://www.rokoko.com

REFERENCES

Yeuhi Abe, Marco Da Silva, and Jovan Popović. 2007. Multiobjective control with frictional contacts. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 249–258.

Eduardo Alvarado, Damien Rohmer, and Marie-Paule Cani. 2022. Generating Upper-Body Motion for Real-Time Characters Making their Way through Dynamic Environments. *Computer Graphics Forum* 41, 8 (2022).

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: data-driven responsive control of physics-based characters. *ACM Transactions On Graphics* 38, 6 (2019), 1–11.

Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. 2018. Physics-Based motion capture imitation with deep reinforcement learning. In *ACM SIGGRAPH Conference on Motion, Interaction and Games*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

Alexander Clegg, Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. 2018. Learning to dress: Synthesizing human dressing motion via deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–10.

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized biped walking control. *ACM Transactions on Graphics* 29, 4 (2010), 130.

Danilo Borges da Silva, Rubens Fernandes Nunes, Creto Augusto Vidal, Joaquim B Cavalcante-Neto, Paul G Kry, and Victor B Zordan. 2017. Tunable robustness: An artificial contact strategy with virtual actuator control for balance. *Computer Graphics Forum* 36, 8 (2017), 499–510.

Marco Da Silva, Yeuhi Abe, and Jovan Popović. 2008. Simulation of human motion data using short-horizon model-predictive control. *Computer Graphics Forum* 27, 2 (2008), 371–380.

Martin De Lasa and Aaron Hertzmann. 2009. Prioritized optimization for task-space control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5755–5762.

Martin De Lasa, Igor Mordatch, and Aaron Hertzmann. 2010. Feature-based locomotion controllers. *ACM Transactions on Graphics* 29, 4 (2010), 1–10.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. *Advances in Neural Information Processing Systems* 30 (2017).

Perttu Hämäläinen, Joose Rajamäki, and C Karen Liu. 2015. Online control of simulated humanoids using particle belief propagation. *ACM Transactions on Graphics* 34, 4 (2015), 1–13.

Tatsuya Harada, Sou Taoka, Taketoshi Mori, and Tomomasa Sato. 2004. Quantitative evaluation method for pose and motion similarity based on human perception. In *IEEE/RAS International Conference on Humanoid Robots*, Vol. 1. 494–512.

Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. 2020. Robust motion in-betweening. *ACM Transactions on Graphics* 39, 4 (2020).

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems* 29 (2016).

Deok-Kyeong Jang, Soomin Park, and Sung-Hee Lee. 2022. Motion Puzzle: Arbitrary Motion Style Transfer by Body Part. *ACM Transactions on Graphics* 41, 3 (2022).

Won-Seob Jang, Won-Kyu Lee, In-Kwon Lee, and Jehee Lee. 2008. Enriching a motion database by analogous combination of partial human motions. *The Visual Computer* 24, 4 (2008), 271–280.

Andrej Karpathy and Michiel Van De Panne. 2012. Curriculum learning for motor skills. In *Canadian Conference on Advances in Artificial Intelligence*. Springer, 325–330.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Taesoo Kwon and Jessica K Hodgins. 2010. Control systems for human running using an inverted pendulum model and a reference motion capture sequence. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 129–138.

Taesoo Kwon and Jessica K Hodgins. 2017. Momentum-mapped inverted pendulum models for controlling dynamic human motions. *ACM Transactions on Graphics* 36, 1 (2017), 1–14.

Seunghwan Lee, Phil Sik Chang, and Jehee Lee. 2022. Deep Compliant Control. In *ACM SIGGRAPH 2022 Conference Proceedings*. Association for Computing Machinery.

Seyoung Lee, Sunmin Lee, Yongwoo Lee, and Jehee Lee. 2021. Learning a family of motor skills from a single motion clip. *ACM Transactions on Graphics* 40, 4 (2021), 1–13.

Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. 2019. Scalable muscle-actuated human simulation and control. *ACM Transactions on Graphics* 38, 4 (2019), 1–13.

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-driven biped control. *ACM Transactions on Graphics* 29, 4 (2010), 129.

Jae Hyun Lim and Jong Chul Ye. 2017. Geometric GAN. *arXiv preprint arXiv:1705.02894* (2017).

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. 2020. Character controllers using motion vaes. *ACM Transactions on Graphics* 39, 4 (2020), 40–1.

Libin Liu and Jessica Hodgins. 2018. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics* 37, 4 (2018), 1–14.

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), 1–14.

Libin Liu, KangKang Yin, and Baining Guo. 2015. Improving sampling-based motion control. *Computer Graphics Forum* 34, 2 (2015), 415–423.

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* 31, 6 (2012), 154–1.

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. In *ACM SIGGRAPH 2010 papers*. 1–10.

Adriano Macchietto, Victor Zordan, and Christian R Shelton. 2009. Momentum control for balance. In *ACM SIGGRAPH 2009 papers*. 1–8.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. 2021. Isaac Gym: High performance GPU-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470* (2021).

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*.

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. *arXiv preprint arXiv:1707.02201* (2017).

Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. 2020. Catch & Carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics* 39, 4 (2020), 39–1.

Igor Mordatch and Emo Todorov. 2014. Combining the benefits of function approximation and trajectory optimization. In *Robotics: Science and Systems*, Vol. 4.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics* 31, 4 (2012), 1–8.

Uldarico Muico, Yongjoon Lee, Jovan Popović, and Zoran Popović. 2009. Contact-aware nonlinear control of dynamic characters. In *ACM SIGGRAPH 2009 papers*. 1–9.

Ofir Nachum, Michael Ahn, Hugo Ponte, Shixiang Gu, and Vikash Kumar. 2019. Multi-agent manipulation via locomotion using hierarchical sim2real. *arXiv preprint arXiv:1908.05224* (2019).

Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics* 38, 6 (2019), 1–11.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics* 37, 4 (2018), 1–14.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics* 36, 4 (2017), 1–13.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. 2019. MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies. *Advances in Neural Information Processing Systems* 32 (2019), 3686–3697.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. 2022. ASE: Large-Scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters. *ACM Transactions on Graphics* 41, 4 (2022).

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics* 40, 4 (2021).

Avinash Ranganath, Pei Xu, Ioannis Karamouzas, and Victor Zordan. 2019. Low dimensional motor skill learning using coactivation. In *ACM SIGGRAPH Conference on Motion, Interaction and Games*. 1–10.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Asako Soga, Yuho Yazaki, Bin Umino, and Motoko Hirayama. 2016. Body-part motion synthesis system for contemporary dance creation. In *ACM SIGGRAPH 2016 Posters*. 1–2.

Kwang Won Sok, Manmyung Kim, and Jehee Lee. 2007. Simulating biped behaviors from human motion data. In *ACM SIGGRAPH 2007 papers*. 107–es.

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural State Machine for Character-Scene Interactions. *ACM Transactions on Graphics* 38, 6, Article 209 (2019).

Sebastian Starke, Yiwei Zhao, Fabio Zinno, and Taku Komura. 2021. Neural animation layering for synthesizing martial arts movements. *ACM Transactions on Graphics* 40, 4 (2021), 1–16.

Jie Tan, Karen Liu, and Greg Turk. 2011. Stable proportional-derivative controllers. *IEEE Computer Graphics and Applications* 31, 4 (2011), 34–44.

Jeff KT Tang, Howard Leung, Taku Komura, and Hubert PH Shum. 2008. Emulating human perception of motion similarity. *Computer Animation and Virtual Worlds* 19, 3-4 (2008), 211–221.
Yuval Tassa, Tom Erez, and Emanuel Todorov. 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 4906–4913.
Yuval Tassa, Nicolas Mansard, and Emo Todorov. 2014. Control-limited differential dynamic programming. In *IEEE International Conference on Robotics and Automation*. 1168–1175.
Hado P van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. 2016. Learning values across many orders of magnitude. *Advances in Neural Information Processing Systems* 29 (2016).
Kevin Wampler, Zoran Popović, and Jovan Popović. 2014. Generalizing locomotion style to new animals with inverse optimal regression. *ACM Transactions on Graphics* 33, 4 (2014), 1–11.
Tingwu Wang, Yunrong Guo, Maria Shugrina, and Sanja Fidler. 2020. Unicon: Universal neural controller for physics-based character motion. *arXiv preprint arXiv:2011.15119* (2020).
Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics* 39, 4 (2020), 33–1.
Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2021. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Transactions on Graphics* 40, 4 (2021), 1–11.
Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2022. Physics-based character controllers using conditional VAEs. *ACM Transactions on Graphics* 41, 4 (2022), 1–12.
Jungdam Won, Jungnam Park, and Jehee Lee. 2018. Aerobatics control of flying creatures via self-regulated learning. *ACM Transactions on Graphics* 37, 6 (2018), 1–10.
Chun-Chih Wu and Victor Zordan. 2010. Goal-directed stepping with momentum control. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 113–118.
Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. 2020. ALLSTEPS: Curriculum-Driven Learning of Stepping Stone Skills. *Computer Graphics Forum* 39 (2020), 213–224.
Pei Xu and Ioannis Karamouzas. 2021. A GAN-Like Approach for Physics-Based Imitation Learning and Interactive Character Control. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 4, 3 (2021).
Zeshi Yang and Zhiqi Yin. 2021. Efficient hyperparameter optimization for physics-based character animation. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 4, 1 (2021), 1–19.
Yuho Yazaki, Asako Soga, Bin Umino, and Motoko Hirayama. 2015. Automatic composition by body-part motion synthesis for supporting dance creation. In *International Conference on Cyberworlds*. IEEE, 200–203.
Yuting Ye and C Karen Liu. 2010a. Optimal feedback control for character animation using an abstract model. In *ACM SIGGRAPH 2010 papers*. 1–9.
Yuting Ye and C Karen Liu. 2010b. Synthesis of responsive motion using a dynamic model. In *Computer Graphics Forum*, Vol. 29. 555–562.
KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics* 26, 3 (2007), 105–es.
Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. 2021. The surprising effectiveness of PPO in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955* (2021).
Wenhao Yu, Greg Turk, and C Karen Liu. 2018. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics* 37, 4 (2018), 1–12.
Victor Zordan, David Brown, Adriano Macchietto, and KangKang Yin. 2014. Control of rotational dynamics for ground and aerial behavior. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1356–1366.
Victor Zordan and Jessica K Hodgins. 2002. Motion capture-driven simulations that hit and react. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 89–96.

A STATE AND ACTION REPRESENTATION

Given the definition in Eq. 4, we have the character state vector $\mathrm{s}_t \in \mathbb{R}^{(n+1)\times N_{\text{link}}\times 13}$, which includes all body links' positions, orientations, and linear and angular velocities of the simulated character in the last $n+1$ frame from $t-n$ to t. To ignore the global coordinate, we assume that the ground height is 0, and all the body links' states are localized based on the position and heading direction of the character's root link (pelvis) at the last frame t. Similarly, if a goal state g_t is provided, we localize the position and direction state in g_t using the same coordinate system with s_t. During our experiments, if multiple goal-directed tasks are involved, we simply concatenate goal states from all the tasks together as the representation of g_t. We refer to the Appendix in the supplementary material for the representation of g_t in our designed goal-directed tasks.

The action a_t is a set of target postures fed into the PD servo. Therefore, we have $\mathrm{a}_t \in \mathbb{R}^{N_{\text{dof}}}$ where N_{dof} is the total degrees of freedom (DoF) in the character model. a_t is assumed to be normalized by the valid movement range of each DoF but without upper and lower bounds applied. The observation space o_t^i for discriminators is similar to s_t. However, we keep only body links' positions and orientations, and the discriminators rely on the pose trajectory of o_t to ensure that the visual velocities between two frames are consistent with the reference motions. As such, we have $\mathrm{o}_t^i \in \mathbb{R}^{(n_i+2)\times N^i_{\text{link}}\times 7}$ where n_i+2 is the number of observed frames as defined in Eq. 3. o_t^i is localized depending on its characteristics. For lower body parts, their motions often involve the character's spatial movement. Therefore, we follow the definition of s_t, and use a local coordinate defined by the root pose at the last observed frame. For upper-body motions, however, we typically care more about the body parts' local poses related to a specific parent body link. Therefore, we use a framewisely defined local system based on the parent link's pose such that the global-space displacement and rotation controlled by the lower body are ignored. In our implementation, for upper-body motions, we choose pelvis as the parent link; and for arm only motions, we choose torso as the parent.

The observation sampled from the reference motions, i.e. $\tilde{\mathrm{o}}_t^i$, is defined the same as o_t^i. However, instead of performing sampling at a fixed frame rate identical to the control policy's working frequency (30Hz in Fig. 2), we do sampling with dynamic interval $\Delta t = \beta T$ where $T = 1/30s$ is the time interval between two frames during simulation and $\beta \sim \textsc{Uniform}(0.8, 1.2)$. In such a way, we scale the reference motion temporally within a small range, for better combining motions from multiple reference sources with inconsistent pace. To keep the motion stable, Δt differs among multiple times of sampling but is identical for the n_i+2 frames of one sample.

B TASK ENVIRONMENT SETUP

B.1 Task: Target Heading

The goal-directed reward is defined as

$$r_t = \langle \dot{\mathrm{x}}^{\text{root}}_{t+1}/||\dot{\mathrm{x}}_t||, \mathrm{g}_t\rangle, \tag{12}$$

where $\dot{\mathrm{x}}^{\text{root}}_{t+1}$ is the horizontal displacement of the character's root link from the frame t to $t+1$. The goal state $\mathrm{g}_t \in \mathbb{R}^2$ is a unit vector representing the target heading direction, which is randomly sampled every 30 frames ($1s$).

B.2 Task: Target Location

The goal-directed reward is defined as

$$r_t = \begin{cases} \exp(-3||\dot{\mathrm{x}}^{\text{root}}_{t+1}/T - \mathrm{v}^*_t||^2/||\mathrm{v}^*_t||^2) & \text{if } ||\mathrm{x}_{t+1} - \mathrm{p}_{\text{goal}}|| > R \\ 1 & \text{otherwise,} \end{cases} \tag{13}$$

where $R = 0.5$ is the goal radius of the target location, $T = 1/30s$ is the time interval between two frames, $\dot{\mathrm{x}}^{\text{root}}_{t+1}/T$ denotes the horizontal velocity of the character's root link from the frame t to $t+1$,

and v_t^* is the target velocity with a preferred speed and a direction toward the target goal location.

The goal state $\mathrm{g}_t \in \mathbb{R}^4$ includes a 2D unit vector representing the direction to the target location, the horizontal distance from the character to the goal, i.e. $||\mathrm{x}_t^{\text{root}} - \mathrm{p}_{\text{goal}}||$, and the preferred speed $||\mathrm{v}_t^*||$. The preferred speed is sampled from $[1, 1.5]$ in the unit of m/s for crouching and walking motions, and from $[1, 3]$ for running. The goal direction is sampled from $[0, 2\pi)$. A timer variable is sampled from $[3, 5]$ in the unit of s for crouching and walking motions, and from $[2, 3]$ for running. We use these three goal variables to obtain the target location. As such, we can perform speed control during the location targeting.

B.3 Task: Aiming

The goal-directed reward is defined as

$$r_t = \begin{cases} \exp(-2||\mathrm{d}_t^{\text{forearm}} - \mathrm{g}_t||^2) & \text{if aiming is activated} \\ \textsc{Clip}(\langle \mathrm{d}_t^{\text{forearm}}, \mathrm{u}^{\text{ref}} \rangle, 0, 0.8)/0.8 & \text{otherwise} \end{cases} \tag{14}$$

where $\mathrm{d}_t^{\text{forearm}} \in \mathbb{R}^3$ is a unit vector representing the direction of the right forearm from the elbow to the hand, and u^{ref} is a unit vector representing the up axis of the world space. In our implementation, the toy pistol is fixed on the right hand, which is linked to the right forearm with a fixed joint. Therefore, we use the direction of the right forearm as the aiming direction. When the aiming action is not activated, we use the 2nd reward term to encourage the character to lift its arm and hold the gun up without aiming anything.

The goal state $\mathrm{g}_t \in \mathbb{R}^3$ is a unit vector representing the target aiming direction. We let $\mathrm{g}_t = 0$ if the aiming action is not activated. When combined with the target location task, aiming is deactivated if the character is close to the goal, i.e. $||\mathrm{x}_t - \mathrm{p}_{\text{goal}}|| \leq R$. g_t is sampled with an elevation angle in range of $[0, \pi/6]$ and azimuth angle in $[0, \pi/4]$.

B.4 Task: Tennis Swing

The goal-directed reward is defined as

$$r_t = \begin{cases} 1.2 + ||\mathrm{v}_{\text{out}}||/10 & \text{if ball was hit and } d_{\text{fall}} = 0 \\ r_t^{\text{pose}} + 0.5\exp(-0.1 d_{\text{fall}}^2) & \text{if ball was hit but } d_{\text{fall}} > 0 \\ r_t^{\text{pose}} & \text{otherwise} \end{cases} \tag{15}$$

where

$$\begin{aligned} r_t^{\text{pose}} &= 0.2 r_t^{\text{shoulder}} + 0.5 r_t^{\text{racket}}, \\ r_t^{\text{shoulder}} &= \exp(-\max(||\mathrm{p}_t^{\text{shoulder}} - \mathrm{p}_t^{\text{ball}}|| - 1, 0)^2), \\ r_t^{\text{racket}} &= \exp(-5||\mathrm{p}_t^{\text{racket}} - \mathrm{p}_t^{\text{ball}}||^2). \end{aligned} \tag{16}$$

$\mathrm{p}_t^{\text{shoulder}}$ is position of the character's right shoulder and $\mathrm{p}_t^{\text{racket}}$ is the position of the racket. To emulate the tennis court, we consider a valid ball falling region with dimension $12m \times 11m$, which is $6m$ ahead of the initial position of the tennis ball along the x-axis. d_{fall} is the distance from the ball's falling point to this region. We let $d_{\text{fall}} = 0$ if the ball will fall or fell in the target region. d_{fall} is estimated by a simple projectile model based on the linear velocity of the ball without considering any friction or air resistance, but updated at every simulation step in order to get an accurate estimation. $||\mathrm{v}_{\text{out}}||$ is the outgoing speed of the tennis ball when it was hit. The purpose of using r_t^{shoulder} is to encourage the character to approach the tennis ball but not necessarily when the distance is less than $1m$ such that the character can have enough space to swing the racket, rather than keeping moving close to the ball.

The goal state $\mathrm{g}_t \in \mathbb{R}^4$ includes a 3D vector representing the position of the ball $\mathrm{p}_t^{\text{ball}}$, and a scalar identifying the heading direction of the character's root link. The heading direction in g_t is used to identify the direction of x-axis toward which the ball is expected to be hit. We let $\mathrm{p}_t^{\text{ball}} = 0$ when constructing g_t if the ball was hit.

B.5 Juggling

The goal-directed reward is defined as

$$r_t = 0.5 r_t^{\text{hand,left}} + 0.5 r_t^{\text{hand,right}} \tag{17}$$

where $r_t^{\text{hand,left}}$ and $r_t^{\text{hand,right}}$ are defined identically but evaluate the performance of the left hand and right hand respectively. For each hand-related reward, we define

$$r_t^{\text{hand}} = \begin{cases} r_t^{\text{throw}} & \text{if } t \bmod \tau = 0 \\ 0.1 r_t^{\text{height}} + 0.9 r_t^{\text{distance}} & \text{otherwise} \end{cases} \tag{18}$$

where τ is the time interval between two trials of ball throwing and

$$\begin{aligned} r_t^{\text{throw}} &= \exp(-5(v_t^{\text{ball}}/V^{\text{ball}} - 1)^2), \\ r_t^{\text{height}} &= \exp(-20(h^{\text{ball}} - h_t^{\text{hand}})^2), \\ r_t^{\text{distance}} &= 0.9\exp(-20 d_t^2) + 0.2\exp(-d_t^2). \end{aligned} \tag{19}$$

As stated in Section 6.4, we employ an automatic catch-and-throw mechanism where a ball is considered caught by a hand and is fixed to that hand if it is close enough, and will be detached (thrown) automatically at a fixed time interval τ between two trials of throwing. The target ball for a hand is decided using a *cascade* juggling pattern. In the reward function, r_t^{throw} measures the performance of ball throwing and is computed only at the frame where a ball is thrown. v_t^{ball} is the vertical velocity of the thrown ball and V^{ball} is the preferred vertical thrown velocity. The preferred velocity is obtained by assuming that the thrown ball will be caught at the same height where it is thrown and at a dwell time t_d before the next time the catching hand performs a thrown. In our experiment, we set $\tau = 2/3s$ (20 frames) with a preferred dwell time $t_{\text{d}} = 0.4s$ (12 frames) and set the number of balls $N_{\text{ball}} = 3$. Given the gravity $g = 9.81m/s^2$, this leads to a preferred velocity

$$V^{\text{ball}} = 0.5g(\frac{\tau}{2}N_{\text{ball}} - t_{\text{d}}) = 2.94m/s. \tag{20}$$

The height-related reward term r_t^{height} measures the error between the hand's vertical position (h_t^{hand}) and the target ball's height when it was thrown (h^{ball}). It encourages the control policy to throw and catch a ball at the same height. We let $r_t^{\text{height}} = 1$ if the target ball was caught by the hand already. The distance-related reward r_t^{distance} measures the distance error between the hand and the target ball. We estimate the ball's vertical movement trajectory using a simple projectile model taking into account only the ball's vertical linear velocity and gravity. The distance d_t is defined as the distance between the hand and the target ball if the hand is above the

Table 3. Hyperparameters

Parameter	Value
policy network learning rate	5×10^{-6}
critic network learning rate	1×10^{-4}
discriminator learning rate	1×10^{-5}
reward discount factor (γ)	0.95
GAE discount factor (λ)	0.95
surrogate clip range (ϵ)	0.2
gradient penalty coefficient (λ^{GP})	10
number of PPO workers (simulation instances)	512
PPO replay buffer size	4096
PPO batch size	256
PPO optimization epochs	5
discriminator replay buffer size	8192
discriminator batch size	512

estimated trajectory, i.e. when the hand is unable to catch the ball at the current hand height, or just the horizontal distance otherwise. As such, r_t^{distance} ignores the vertical ball-hand distance if the hand is able to catch the ball at its current height, and thus prevents the hand from aggressively moving toward the ball vertically.

The goal state $g_t \in \mathbb{R}^{19}$ includes the three balls' states (position and linear velocity) and a timer variable counting the time left before the next throwing of the ball by one hand. The ball states are in the order of the left-hand target ball, the right-hand target ball, and the other ball. For a caught target ball, we let the corresponding state be zero.

C HYPERPARAMETERS

The hyperparameters used for policy training is listed in Table 3. Half of the samples for discriminator training are from the simulated character and half are sampled from the reference motions. The character state horizon $n + 1$ is chosen as 4, and the discriminator observation horizon $n_i + 2$ is 3 for aiming motions and 5 for other motions. The objective weight ω_k in Eq. 9 is 0.5 shared equally by all goal-related objectives. In the Juggling with Target Location task, given the difficulty of ball catching, the juggling task is assigned a weight of 0.6, the locomotion task has a weight of 0.1, and the imitation tasks account for the remaining weight with a ratio of $1 : 4$ for juggling and walking motion imitation. In the Aiming+Locomotion task, the upper-body motion of aiming has a weight of 0.2 and the lower-body motion has a weight of 0.3. On the other tests, besides the weights taken by the goal-related objectives, the remaining weight is shared equally by the imitation objectives.