Control Strategies for Physically Simulated Characters Performing Two-player Competitive Sports

JUNGDAM WON, Facebook AI Research
DEEPAK GOPINATH, Facebook AI Research
JESSICA HODGINS, Facebook AI Research

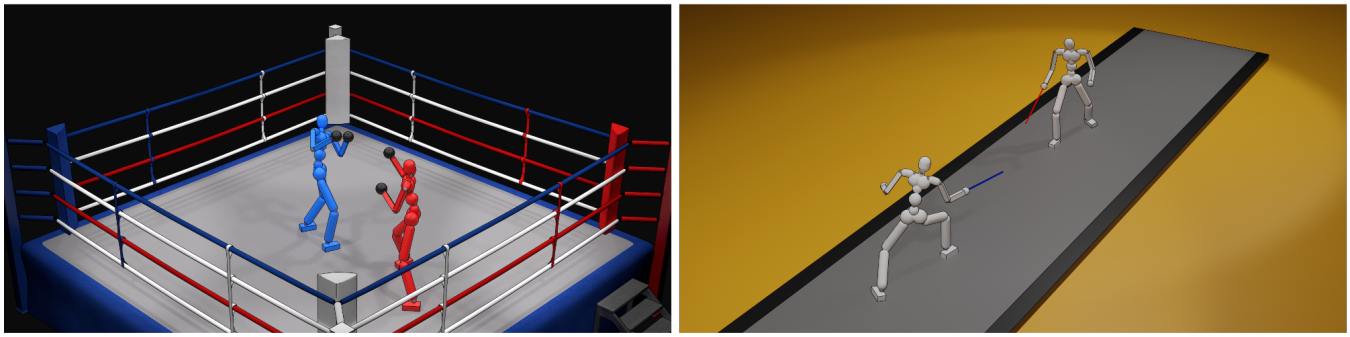


Fig. 1. Characters performing two-player competitive sports such as boxing (left) and fencing (right) using learned control strategies.

In two-player competitive sports, such as *boxing* and *fencing*, athletes often demonstrate efficient and tactical movements during a competition. In this paper, we develop a learning framework that generates control policies for physically simulated athletes who have many degrees-of-freedom. Our framework uses a two step-approach, learning basic skills and learning bout-level strategies, with deep reinforcement learning, which is inspired by the way that people how to learn competitive sports. We develop a policy model based on an encoder-decoder structure that incorporates an autoregressive latent variable, and a mixture-of-experts decoder. To show the effectiveness of our framework, we implemented two competitive sports, *boxing* and *fencing*, and demonstrate control policies learned by our framework that can generate both tactical and natural-looking behaviors. We also evaluate the control policies with comparisons to other learning configurations and with ablation studies.

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

Additional Key Words and Phrases: Character Animation, Physics-based Simulation and Control, Reinforcement Learning, Deep Learning, Neural Network, Multi-agent

ACM Reference Format:
Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2021. Control Strategies for Physically Simulated Characters Performing Two-player Competitive Sports. *ACM Trans. Graph.* 40, 4, Article 1 (August 2021), 11 pages. https://doi.org/10.1145/3450626.3459761

Authors' addresses: Jungdam Won, Facebook AI Research; Deepak Gopinath, Facebook AI Research; Jessica Hodgins, Facebook AI Research.

0730-0301/2021/8-ART1
https://doi.org/10.1145/3450626.3459761

1 INTRODUCTION

Many competitive sports involve long periods of routine play interspersed with occasional dramatic demonstrations of agility. Those strategic moments are often what determine the outcome of the competition and spectators wait and cheer for those highlights. But both the routine play and the scoring moments are hard to reproduce automatically in animated characters because of the complexities involved in the interactions between the competing athletes. If we had the ability to create virtual athletes who could automatically perform all the movements of their sports and assemble them to develop a winning strategy, that functionality would open up many new applications in computer games, commercial films, and sports broadcasting.

Creating animated scenes with multiple people is challenging because it requires not only that each individual behave in a natural way but that their interactions with each other are synchronized in both the temporal and spatial domains to appear natural. The denser the interactions are, the more challenging the problem is as there is no time to "reset" between interactions. Using physically simulated characters simplifies one part of the problem because low-level physical interactions such as collision are automatically generated through simulation. However, coordinating the different skills such as jabs and punches or thrusts and parries or the bout-level strategies of countering and pressure-fighting has not been studied in depth because of the computational complexity of learning the series of skills that comprise a full competition. A key challenge in using the simulated characters for competitive sports is that we need to learn both the basic skills and bout-level strategies so that they work properly in concert.

In recent years, deep reinforcement learning techniques have shown promising results in creating controllers or control policies for physically simulated humanoids for common behaviors such as

locomotion and manipulation as well as more esoteric behaviors such as bicycle riding and gymnastics. Most of these behaviors involve only a single character and behaviors that require interactions among the characters have not been studied in depth.

In this paper, we explore techniques for training control systems for two-player competitive sports that involve physical interaction. We develop a framework that generates control policies for this scenario, where the humanoids have many degrees-of-freedom and are actuated by joint torques. Our framework is inspired by the way that people learn how to play competitive sports. For most sports, people first learn the basic skills without an opponent and then learn how to combine and refine those skills by competing against an opponent. We mimic these two processes, learning basic skills and learning bout-level strategies, with deep reinforcement learning. We develop a policy model based on an encoder-decoder structure that incorporates an autoregressive latent variable, and a mixture-of-experts decoder. To show the effectiveness of our framework, we implemented two competitive sports, *boxing* and *fencing*, and demonstrate control policies learned by our framework that can generate both responsive and natural-looking behaviors for the players. To evaluate our framework, we compare with other plausible design choices and use ablation studies to understand the contribution of individual components.

The contributions of this paper are as follows:

- **Novel Results.** We demonstrate successful control policies that generate both responsive and natural-looking behaviors in competitive settings for high degree-of-freedom physically simulated humanoids. This problem has not previously been studied in depth.
- **Policy Model and Learning Procedure.** Our policy model is designed for efficient transfer learning which enables us to use a few motion clips captured with a single actor only. For example, it generates plausible competitive policies for boxing by using just minutes of a boxer practicing alone.
- **Baseline for Future Research** We develop new competitive environments, boxing and fencing, where two physically simulated players can fight either with their fists or with swords to win the game. The environments were selected because of the need for physical interactions. To support future researchers, we plan to share our environments and learned policies.

2 RELATED WORK

There is a long history in computer animation of developing control systems for physically simulated characters. We review the papers that are most closely related to our work because they handle multiple characters or physically simulated characters. We also review several studies of reinforcement learning techniques that are similar to the ones we employ and focus on their applications to character animation and multi-agent problems.

2.1 Multi-character Animation

Many researchers have proposed techniques for creating realistic multi-character animation. Most of the existing techniques are kinematics-based and physics is not considered in generating the animation.

A popular approach is patch-based generation. The key idea is to build short sequences of motion containing interactions which can then be glued together to create longer sequences with intermittent interactions. Because the patches are usually pre-built with motion capture data involving real interactions, we can ensure that the generated scenes will always have plausible interactions. This approach was first introduced in [Lee et al. 2006], and the idea was then extended to two character interaction [Shum et al. 2008], and crowd scenes [Yersin et al. 2009]. Several methods have been developed to edit the details of the scene [Ho et al. 2010; Kim et al. 2009] and to make bigger and more complex scenes [Henry et al. 2014; Hyun et al. 2013; Kim et al. 2014; Kwon et al. 2008; Won et al. 2014] using the patches as building blocks.

There have also been approaches that directly synthesize multi-character animation without using pre-defined motion patches, instead imposing soft or hard constraints on the interactions. For example, given a motion graph constructed from boxing motions captured from a single person, a kinematic controller that approaches and hits the opponent can be pre-computed by dynamic programming where the controller is represented by a transition table [Lee and Lee 2004]. Liu et al. [2006] created multi-character motions by solving a space-time optimization problem with physical and user-specified constraints. Shum and his colleagues proposed an online method based on game tree expansion and min-max tree search, where competitive and collaborative objectives exist simultaneously [Shum et al. [n.d.], 2007; Shum et al. 2012]. Kwon et al. [2008] tackled a similar problem by learning a motion transition model based on a dynamic Bayesian network on top of coupled motion transition graphs. Wampler et al. [2010] demonstrated a system that can generate feints or misdirection moves in two-player adversarial games based on game theory, where stochastic and simultaneous decision allows such nuanced strategies based on unpredictability. Our method belongs to this class of approaches because we directly synthesize multi-character animation by using only individually captured motions (with no opponent present). However, we incorporate dynamic simulation to ensure the naturalness of impacts and other physical effects.

2.2 Physics-based Character Animation

Physics-based character animation involves characters that are modeled as interconnected rigid bodies with mass and moment of inertia and controlled by joint torques or muscle models. The enforcement of physics laws prevents motions that are physically unrealistic but do not prevent unnatural motions such as those in which response times are too fast or joints are too strong. Many different approaches have been proposed to supplement the constraints of physics with additional control, learning, or optimization to produce natural motion for humanoid characters. Although physics-based methods has shown promising results for individual characters performing a wide variety of behaviors, there exist only a few studies for multi-character animations. Zordan et al. [2002; 2005] proposed motion capture-driven simulated characters that can react to external perturbations, where two-player interactions such as fighting,

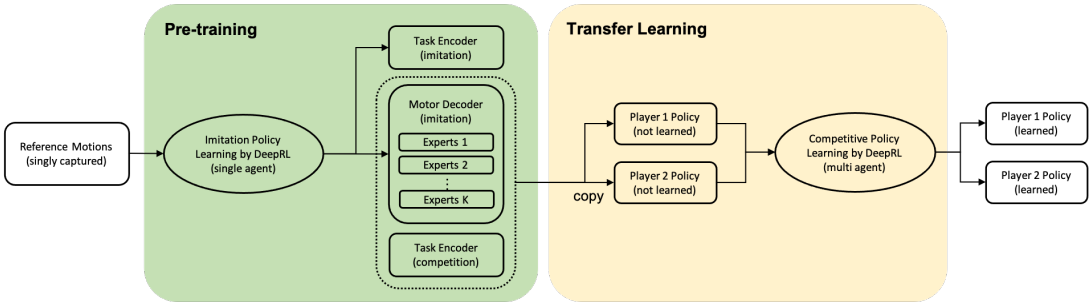


Fig. 2. Overview of the system.

boxing, table-tennis, and fencing were shown as examples. Recently, deep reinforcement learning has shown ground breaking results for physically simulated characters performing behaviors such as locomotion [Berseth et al. 2018; Peng et al. 2017; Yu et al. 2018], such as imitation [Bergamin et al. 2019; Chentanez et al. 2018; Lee et al. 2019; Merel et al. 2017; Peng et al. 2018; Wang et al. 2017; Won et al. 2020; Won and Lee 2019], and such as other skills [Clegg et al. 2018; Liu and Hodgins 2017, 2018; Xie et al. 2020]. However, the number of studies on multi-characters is still limited. Park and colleagues [2019] showed an example of *chicken hopping* where two players hop on one leg and bump into each other to knock their opponent over. Although the paper demonstrated physically plausible multi-character animations, the characters in the scenes do not have high-level strategies (intelligence) for fighting. Haworth and colleagues [2020] demonstrated a crowd simulation method for physically simulated humanoids based on multi-agent reinforcement learning with a hierarchical policy that is composed of navigation, foot-step planning, and bipedal walking skills. Unlike the previous approaches, this approach learns control policies that can adapt to interactions among multiple simulated humanoids. However, only limited behaviors such as navigation and collision-avoidance were demonstrated. Our work focuses on generating more complex and subtle interactions automatically for competitive environments, which is one of the challenging problems in multi-character animation.

2.3 Multi-agent Reinforcement Learning

We solve a multi-agent reinforcement learning (RL) problem that learns optimal policies of multiple agents, where the agents interact with each other to achieve either a competitive or a cooperative goal. This research topic has a long history in machine learning, and we recommend readers to [Busoniu et al. 2008; Hu and Wellman 1998; Nguyen et al. 2020] for a general introduction and details.

The multi-agent problem is much more challenging than the single-agent problem because of the difficulty of learning with a non-stationary (moving target). All agents update their policies simultaneously, so models of the policies of other agents can become invalid after just one learning iteration. In multi-agent RL, there are two types of problems: cooperative and competitive. In the cooperative problem, there exists a common goal that the agents need to achieve together, so communication channels among the agents are commonly included in the policy model [Claus and Boutilier 1998; Lowe et al. 2017; OroojlooyJadid and Hajinezhad 2020]. In the competitive problem, each agent has their own goal that is incompatible with those of the other agents, so communication channels are usually ignored and each agent performs autonomously with only observations and models of their opponent's behavior [Bai and Jin 2020; Baker et al. 2019; Bansal et al. 2018; Seldin and Slivkins 2014].

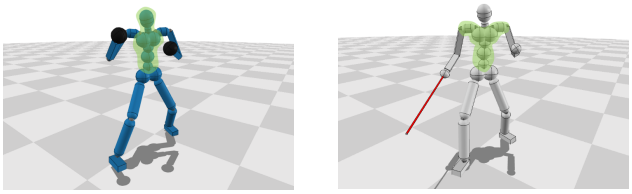
Recently, Lowe et al. [2017] proposed an algorithm called MADDPG that solves the multi-agent RL problem as if it were a single-agent RL problem by using a shared Q-function across all the agents. Bansal et al. [2018] showed control policies for physically simulated agents, which emerged from a competitive environment setting. Baker et al. [2019] demonstrated the evolution of policies in a hide-and-seek game, where the policies were learned with simple competitive rewards in a fully automatic manner. Although the last two results showed the potential of multi-agent RL in generating multi-character motions, there exists room for improvement: the characters were very simple, the motions were not human-like, or careful development of policies by curriculum learning was required.

3 OVERVIEW

Our framework takes a set of motion data that includes the basic skills of a two-player competitive sport as input and generates control policies for two physically simulated players. The control policies allow the players to perform a sequence of basic skills with the right movements and timing to win the match. Figure 2 illustrates the overview of the framework. First, we collect a few motion clips that include the basic skills of the sport performed without an opponent. A single imitation policy is then learned for the motions by using single-agent deep reinforcement learning. Finally, we transfer the imitation policy to competitive policies for the players where each player enhances their own policy by multi-agent deep reinforcement learning with competitive rewards. To effectively transfer from the imitation policy to the competitive policy, we use a new policy model composed of a task-encoder and a motor-decoder, the details will be explained in section 4.

3.1 Environments

We created two competitive sport environments, *boxing* and *fencing*, as our examples, where two players compete with each other in an attempt to win the match (Figure 1). We developed the environments based on publicly available resources from [Won et al. 2020]. In the boxing environment, we enlarged both hands so that the size is



(a) Boxing (b) Fencing

Fig. 3. The boxing and fencing characters. The body parts where points are scored if they are hit are marked in green.

similar to the size of boxing glove. The size of the arena is 5 m by 5 m, and the player is only allowed to hit the upper body of the opponent with their gloves (Figure 3). One round time is 60 s and the player who inflicts the most damage (physical forces) on his opponent wins the match. For the fencing environment, we attached a blade with a weld joint at the end of the right hand to mimic a firmly held sword. The size of the arena is 12 m x 2 m, the player is only allowed to touch the torso of the opponent with the blade, which is similar to the rules of *foil*, a type of fencing. The player who touches their opponent first wins the match, however, the match can become a draw if the player successfully counterattacks within 1 s after the first touch occurs. We control the players in both environments by using a target posture (i.e. a set of target joint angles), then a stable PD controller [2011] computes joint torques for all joints except for the root joint.

3.2 Multi-agent Reinforcement Learning

We formulate our environment as a partially observable Markov game [Littman 1994]. A Markov game with k agents can be described as a tuple $\{\mathcal{S}, \mathcal{A}_1, \cdots, \mathcal{A}_k, \mathcal{O}_1, \cdots, \mathcal{O}_k, \mathcal{R}_1, \cdots, \mathcal{R}_k, \rho, \mathcal{T}\}$, where $\mathcal{S}$ is a set of states, $\mathcal{A}_i$ and $\mathcal{O}_i$ are sets of actions and observed states for i-th agent, respectively, $\mathcal{O}_i$ only includes partial information of the entire game state $\mathcal{S}$ because there exist multiple independent agents. The initial state of the game is sampled from the distribution $\rho : \mathcal{S} \rightarrow [0, 1]$, each agent takes an action $\mathbf{a}_i \in \mathcal{A}_i$ given own observed state $\mathbf{o}_i \in \mathcal{O}_i$, the game proceeds by the state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_k \rightarrow \mathcal{S}$ and each agent receives a scalar reward signal by own reward function $\mathcal{R}_i : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_k \rightarrow \mathbb{R}$. The process repeats until one of the terminal conditions of the game is satisfied. The goal of multi-agent deep reinforcement learning is to find an optimal policy $\pi_i : \mathcal{O}_i \times \mathcal{A}_i \rightarrow [0, 1]$ that maximizes the expected return $\mathbb{E}[\Sigma_{t=0}^{T} \gamma^t r_{i,t}]$ for each agent, where $\gamma \in (0, 1)$ is a discount factor, T is the time horizon, and $r_{i,t}$ is a reward value at time t.

4 METHODS

Figure 2 shows the overall framework of our approach. The policy structure that provides effective transfer learning from the imitation policy to the competitive policy is first described, followed by the mathematical formalism for learning the two policies. For simplicity, we drop the player index i in the mathematical formalism below unless it is explicitly required. Because the same framework can be used for two different sports *boxing* and *fencing* with minimal changes in the state and rewards, we first explain all the technical components with *boxing* as the example and then the extra details for *fencing* are described at the end of the section.

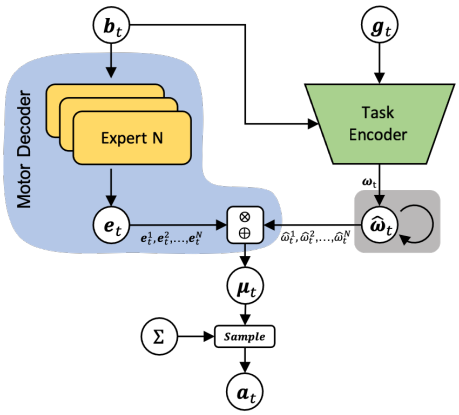


Fig. 4. Our transferable policy model. The model includes the task encoder (green) and the motor decoder (blue) composed of the experts (yellow). The output of task encoder is updated in an autoregressive manner (gray).

4.1 Transferable Policy Structure

In our environment, the default action is a target posture which is represented as a 51 dimensional vector. Learning a control policy from scratch in such high-dimensional space using naïve random exploration and no prior knowledge can easily result in unnatural albeit physically realistic motions. This result is particularly likely when the reward function is under-specified (sparse). Incorporating motion capture data allows us to utilize dense rewards by comparing the data with the motions of the simulated character, however, there are still unsolved challenges in adopting motion capture data for multi-agent environments. First, collecting high-quality motion capture data with multiple actors is much harder than collecting data from a single actor. Accurate capture is especially difficult when we require dense and rich interactions among multiple actors because of occlusions and subtleties in the motion during physical interactions. Including two actors also increases the magnitude of what must be captured to cover all of the needed interaction scenarios. We design our framework to use motion capture data that were captured with a single actor only and create the needed interactions through simulation and learning. The framework uses a two step approach where we first build a single imitation policy with individually captured motions and then transfer the policy into two competitive policies for the players. The motivation for our approach comes from the way that people learn how to play competitive sports. Novice players first learn the basic skills through imitating expert players' demonstrations, and then they refine the learned basic skills and learn tactics while playing against an opponent.

Figure 4 illustrates the policy structure that we use for efficient transfer learning, where the observed state of the player $\mathbf{o}_t = (\mathbf{b}_t, \mathbf{g}_t)$ at time t is composed of the body state (proprioception) $\mathbf{b}_t$ and the task-specific state $\mathbf{g}_t$. The structure is composed of a motor decoder and a task encoder. The motor decoder includes a

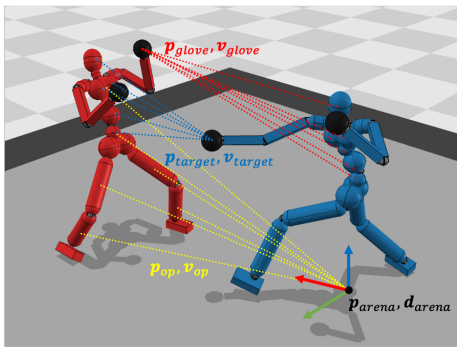


Fig. 5. An illustration of the task-specific state in boxing for the blue player.

pool of N experts where each expert has only the body state as input. The output of the motor decoder is a set of outputs from the experts $\mathbf{e}_t = (\mathbf{e}_t^1, \mathbf{e}_t^2, \cdots, \mathbf{e}_t^N)$. The task encoder receives the whole observed state as input and then generates expert weights $\boldsymbol{\omega}_t = (\omega_t^1, \omega_t^2, \cdots, \omega_t^N)$. The output of the task encoder is used to update the latent expert weights $\hat{\boldsymbol{\omega}}_t$ in an autoregressive manner $\hat{\boldsymbol{\omega}}_t = (1-\alpha)\boldsymbol{\omega}_t + \alpha\hat{\boldsymbol{\omega}}_{t-1}$, where α controls smoothness of the weight change. The mean action $\boldsymbol{\mu}_t$ is computed by the weighted sum $\boldsymbol{\mu}_t = \Sigma_{i=1}^N \hat{\omega}_t^i \mathbf{e}_t^i$, and we sample an action $\mathbf{a}_t$ stochastically from a Gaussian distribution whose mean and covariance are $\boldsymbol{\mu}_t$ and Σ, respectively, with a constant diagonal matrix used for the covariance. When transferring the imitation policy to the competitive policy, only the motor decoder of the imitation policy is reused. The motor decoder and a new task encoder whose input dimension matches to the input competitive sport environment constitute a new policy for each player as illustrated in Figure 2.

4.2 Pre-training: Imitation Policy

To learn a policy that can imitate basic boxing skills, we first used a set of boxing motion clips from the CMU Motion Capture Database [CMU 2002]. The data used is approximately 90 s, with four motion clips each between 10-30 s. We also used their mirrored versions, so the total length is approximately 3 minutes. The actor in the motion capture data performed several skills in an unstructured way. We use this data without any extra post-processing such as cutting them into individual skills or using manual labels such as phase. Instead, we learn a single imitation policy by using deep reinforcement learning (RL) with imitation rewards. The body state is $\mathbf{b}_t = (\mathbf{p}, \mathbf{q}, \mathbf{v}, \mathbf{w})$, where $\mathbf{p}$,$\mathbf{q}$,$\mathbf{v}$, and $\mathbf{w}$ are positions, orientations, linear velocities, and angular velocities of all joints, respectively, which are represented with respect to the current facing transformation of the simulated player.

The task-specific state $\mathbf{g}_t = (\mathbf{b}_{t+1}^{ref}, \mathbf{b}_{t+2}^{ref}, \cdots)$ includes a sequence of future body states extracted from the current reference motion. In the examples presented here, we use two samples 0.05 and 0.15 s in the future. We use the multiplicative imitation reward function and also perform early-termination based on the reward in a similar fashion to [Won et al. 2020].

4.3 Transfer: Competitive Policy

To create a competitive policy, we replace the task encoder with a new task encoder whose input dimension matches the boxing environment and reuse the motor decoder as is. Please note that the experts in the motor decoder do not represent specific individual skills because they are learned simultaneously, rather they can be regarded as basis functions for imitating the input motion capture data. As a result, reusing the motor decoder implies that the action space is a weighted combination of the experts (basis functions) built from the imitation task, which is a greatly reduced space that includes postures existing in the input motions when compared to the entire posture space. A copy of the new policy is initially assigned to each player, and then those policies are refined through multi-agent deep reinforcement learning. Because the goal for the two players in boxing is the same, they can be modeled by using the same formulation. The reward function is

$$r = r_{\text{match}} + w_{\text{close}} r_{\text{close}} + w_{\text{facing}} r_{\text{facing}} - w_{\text{energy}} r_{\text{energy}} - w_{\text{penalty}} \sum_i r^i_{\text{penalty}} \quad (1)$$

where r_{match} generates signals related to the match. In our boxing example, it measures how much the player damaged the opponent and how much the player was damaged by the opponent in the last step. We measure damage by the magnitude of force,

$$r_{\text{match}} = \|f_{pl->op}\| - \|f_{op->pl}\| \quad (2)$$

where $f_{pl->op}$ is the contact normal force that the player applied to the opponent and $f_{op->pl}$ is the force applied by the opponent to the player. When measuring damage, we only consider collisions between both gloves and a subset of the upper bodies: the head, the chest, and the belly (Figure 3). For numerical stability we clip the magnitude of the force by 200N when computing $\|f_{pl->op}\|$ and $\|f_{op->pl}\|$. The close and facing terms increase the probability of interaction while learning policies, $r_{\text{close}} = exp(-3d^2)$ encourages the players to be close where d is the closest distance between the player's gloves and the opponent's upper body and r_{facing} encourages the players to face each other

$$\begin{aligned} r_{\text{facing}} &= exp(-5\|\bar{\mathbf{v}} \cdot \hat{\mathbf{v}} - 1\|) \\ \bar{\mathbf{v}} &= (\hat{\mathbf{p}}_{\text{op}} - \hat{\mathbf{p}}_{\text{pl}})/\|\hat{\mathbf{p}}_{\text{op}} - \hat{\mathbf{p}}_{\text{pl}}\| \end{aligned} \quad (3)$$

where $\hat{\mathbf{v}}$ is the facing direction and $\hat{\mathbf{p}}_{\text{op}}$,$\hat{\mathbf{p}}_{\text{pl}}$ are the facing positions of the opponent and the player, respectively (i.e. the root joint positions projected to the ground). We also add an energy minimization term to make the players move in an energy-efficient manner

$$\begin{aligned} r_{\text{energy}} &= \kappa_{\text{dist}} \sum_j \|\mathbf{a}_j\|^2 \\ \kappa_{\text{dist}} &= 1 - exp(-50\|max(0, l-1)\|^2) \end{aligned} \quad (4)$$

where $\mathbf{a}_j$ is the angular acceleration of i-th joint and l is the horizontal distance between the two players. The energy term has an effect only when the players are more than 1 m apart. This term approximates the behaviors of boxers as they tend to conserve energy by only throwing punches when their opponent is close enough that they might be effective. Without this term, the boxers would throw punches indiscriminately as the punches have no cost. Finally, we

add extra penalty terms to prevent the players from entering undesirable or unrecoverable situations that hinder sample efficiency. Each of these penalty terms is a binary signal:

$$r^i_{\text{penalty}} = \begin{cases} 1 \text{ if the condition is satisfied} \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

We also terminate the current episode if one of the conditions is activated. In our boxing example, we use two conditions: *fall* and *stuck*. The *fall* checks whether any body part of the player except for the foot touches the ground. This penalty is helpful to learning to maintain balance in the early phase of the learning (before tactics emerge) and also for recovering from blows by the opponent. The *stuck* was inspired by real boxing matches where a judge interrupts and restarts the match if the two players are grasping each other in their arms or one player is trapped on the rope and can only defend themselves. The judge restarts the match because there are almost no meaningful actions that can occur when the players are in these configurations. We detect the first situation when the distance between the players is less than $0.4\,m$ for more than $5\,s$ and the second situation when one of the players is located near the ropes for more than $5\,s$.

The observed state of the player is composed of the body state and the task-specific state, where the body state is the same as the one used for learning the imitation policy and the task-specific state includes the current status of the match

$$\mathbf{g}_t = (\mathbf{p}_{\text{arena}}, \mathbf{d}_{\text{arena}}, \mathbf{p}_{\text{op}}, \mathbf{v}_{\text{op}}, \mathbf{p}_{\text{glove}}, \mathbf{v}_{\text{glove}}, \mathbf{p}_{\text{target}}, \mathbf{v}_{\text{target}}) \tag{6}$$

where $\mathbf{p}_{\text{arena}}$, $\mathbf{d}_{\text{arena}}$ are the relative position and the facing direction of the player in the arena. $\mathbf{p}_{\text{op}}$, $\mathbf{v}_{\text{op}}$ are relative positions and velocities of the opponent's body parts, which are represented in a coordinate system aligned with the player's facing direction. $\mathbf{p}_{\text{glove}}$, $\mathbf{v}_{\text{glove}}$ are the relative positions and velocities of the opponent's gloves with respect to the player's upper body parts, and $\mathbf{p}_{\text{target}}$, $\mathbf{v}_{\text{target}}$ are relative positions and velocities of the opponent's upper body parts with respect to the player's gloves (Figure 5). The dimension of the entire observed state is 623 in total.

4.4 Learning

When transferring a pre-trained policy with an encoder and a decoder to a new task, there are two popular approaches. One is to learn only the encoder while the decoder remains fixed (*Enc-only*), the other is to update the entire structure in an end-to-end manner (*Enc-Dec-e2e*). There are pros and cons to both approaches. Both approaches show similar results if the new task is the same or similar to the previous task (pre-training), whereas the *Enc-Dec-e2e* usually shows faster and better adaptation and convergence if the new task is quite different from the original task. For example in the boxing environment, we pre-train our motor decoder in the single-agent setting by imitating motion clips that do not have any interaction with the opponent. The competitive policy, however, needs to learn abilities such as withstanding an attack or punching the opponent. *Enc-only* is not sufficient to learn sophisticated tactics for the competitive environment unless we use motion capture data that covers almost all possible scenarios in pre-training. *Enc-only* is more robust to the *forgetting* problem (where a pre-trained policy easily forgets its original abilities for the original task and adapts only to the new task) because the motor decoder does not change at all. The *forgetting* problem does not matter if the performance on the new task is the only concern, however, it can be problematic for generating human-like motions for the player because a large deviation from the original motor decoder trained on motion capture data can easily lead to a situation where the motor decoder is no longer generating natural motions. To tackle this challenge, we do *Enc-only* in the early stage of transfer learning, then alternate between *Enc-Dec-e2e* and *Enc-only*. This learning procedure enables the players to learn meaningful tactics while preserving the style of the motions existing in the original motion capture data. In our experiments, we start with *Enc-only* for 300-500 iterations, then alternate between them every 50th iteration.

We use the decentralized distributed proximal policy optimization (DD-PPO) algorithm [2020] to learn imitation and competitive policies. As the name implies, DD-PPO is an extended version of the proximal policy optimization (PPO) algorithm, which runs on multiple machines (distributed), requires no master machine (decentralized), and does synchronized update for policy parameters. In the original PPO, training tuples generated on each machine are collected by the master machine, the machine computes gradients for all the training tuples, the current policy is updated by the gradients, and the updated policy parameters are shared over the network by broadcasting. In the DD-PPO, each machine computes the gradients for training tuples that it generated and only the gradients are shared over the network to compute the average gradients. This process reduces the communication and the overhead of computing gradients when the number of training tuples per iteration is large.

4.5 Details for Fencing

Building control policies for the fencing environment requires minimal changes in the state and rewards. For the state, the terms related to the gloves $\mathbf{p}_{\text{glove}}$, $\mathbf{v}_{\text{glove}}$ are replaced by the state of the sword attached to the right hand of the fencer. We also include a trigger vector $(l_{\text{pl}}, l_{\text{op}})$ that represents the history of touch, where l_{pl} becomes 1 if the player touches the opponent and it does not change until the end of the match, otherwise the value is zero, and l_{op} is the opposite. The reward function for fencing is

$$r = r_{\text{match}} + w_{\text{close}} r_{\text{close}} - w_{\text{penalty}} \sum_i r^i_{\text{penalty}} \tag{7}$$

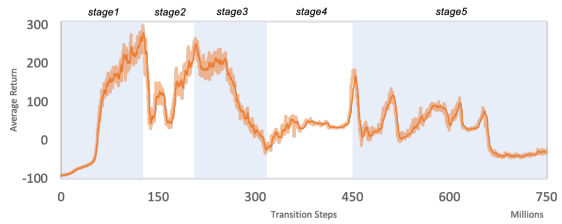
where r_{close} is the same as the boxing environment, r_{match} gives 1 for the winner, -1 for the loser, and 0 when the match becomes a draw. In contrast to that the boxers can collect r_{match} continuously until the end of the match, the fencers can get non-zero r_{match} only once when the match terminates (i.e. when the winner and the loser are decided). We use two penalty terms *fall* and *out-of-arena*. *fall* is the same as in boxing and *out-of-arena* is activated when the fencer goes out of the arena because there is no physical boundary in the fencing environment unlike the boxing environment. We use the same learning algorithm without changing any hyper-parameters.

5 RESULTS

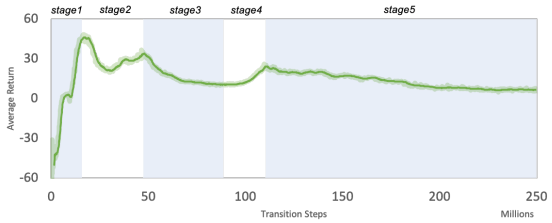
Table 1 includes all parameters used for physics simulation, DD-PPO algorithm, and the reward functions. We used the PyTorch [2019] implementation in RLlib [2018] for the DD-PPO algorithm. We

Table 1. Parameters

Simulation	Simulation rate (Hz)	480
	Control rate (Hz)	30
DD-PPO	Policy learning rate (α_π)	0.00001
	Discount factor (γ)	0.99
	GAE and TD (λ)	0.95
	# of tuples per policy update (T)	150000
	Batch size for policy update (n)	2000
	Iteration for policy update (N)	20
	Clip parameter (ϵ)	0.2
Rewards	w_{close}	0.2
	w_{facing}	0.05
	w_{energy}	0.001
	w_{penalty}	30.0



(a) Boxing.



(b) Fencing.

Fig. 6. Learning curves on the average return.

use 8 and 4 experts for the boxing and the fencing environments, respectively. Each expert in the motor decoder consists of two fully-connected hidden layers (64 nodes, ReLu activation units) followed by one output layer (51 nodes, no activation unit). The task encoder for the boxing environment is made up of a two-layer LSTM (128 hidden size) and a hidden layer (128 nodes, ReLu activation units) followed by one output layer (N nodes, *tanh* activation unit), where N is the number of experts. For the task encoder of the fencing environment, we use two fully-connected hidden layers (128 nodes, ReLu activation units) instead of the LSTM layers because long-term memory is not crucial due to the short length of the match.

In all our experiments, we used 12 machines, where each machine has 2 CPUs (Intel Xeon CPU E5-2698 v4). The pre-trainig (imitation policy learning) takes approximately 1 day (2e8 transition tuples) for both sports, and the transfer learning (competitive policy learning) requires 4-5 days (5e8-7e8 transition tuples) for the boxing environment, 2-3 days (2e8-3e8 transition tuples) for the fencing environments to learn plausible policies. The competitive policy learning is almost 1.5 times slower than the imitation policy learning because the environment includes two simulated characters for which both computing stable PD control and evaluating control policies are required, respectively.

5.1 Competitive Policy

We learn competitive policies for the boxing and the fencing environments. Figure 11 depicts two boxers fighting each other. The boxer moves around with light steps when the opponent is far away, keeps the opponent in check by using light jabs in the middle distance (slightly farther than the reach). When the opponent is entirely within reach, the boxer punches the opponent repeatedly while blocking or avoiding the opponent's attack. If the boxer is held in a corner, the boxer sometimes tries to reverse the situation by using the ropes in the arena. These examples and others shown in the accompanying video demonstrate that our approach successfully learns plausible tactics for a boxing environment while preserving the style of the original motions.

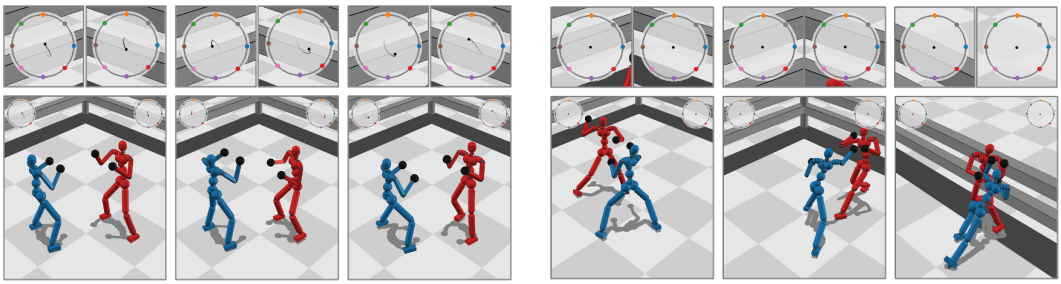
Figure 12 depicts an example of the fencing motion (animations are in the accompanying video), where each row shows different match results. The matches typically end within 5 s, which is similar to actual human matches. In our experiments, one fencer typically learns aggressive tactics while the other fencer learns to stay in place and play defensively. Due to Gaussian noise in the action, the outcome of the match changes every match.

5.2 Learning Curves

The progress of learning does not increase monotonically in competitive multi-agent RL environments as it often does in single-agent RL or cooperative multi-agent RL environments. A similar phenomenon is also observed in learning a generative adversarial network (GAN) [Kod [n.d.]]. What mostly drives the entire learning process is the match reward, which has the property of a zero-sum game. Figure 6a illustrates a learning curve of the average return for the boxing environment. The learning process can be described in five stages: learning to stand in place (stage 1), wandering around while unintended collisions happen (stage 2), development of punches (stage 3), development of blocks and avoidance (stage 4), repeated fluctuation between stage 3 and stage 4 (stage 5). The fencing environment also shows the five stages during learning except that there is less fluctuation in the 5th stage (Figure 6b). In the case of two-player zero sum games, it is referred to as *convergence* when both players behave in a regret-minimizing manner (i.e. the player tries to minimize loss when the opponent maximizes their own benefit). In our experiments, we judge that the policies converge when the match reward is around zero during stage 5.

5.3 Evaluation

5.3.1 Learning Procedure. As mentioned in section 4.4, our learning procedure for the task encoder and the motor decoder is crucial to build policies that generate human-like motions. We compare our learning procedure of updating the task encoder first and then updating the entire structure later with two alternative methods:

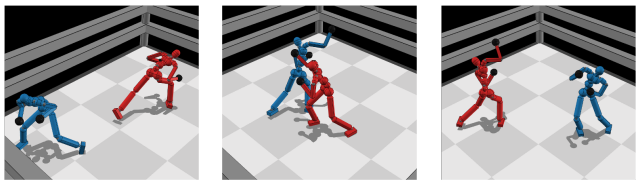


(a) Enc-only (b) Enc-Dec-e2e

Fig. 7. Other learning procedures. *Enc-Dec-e2e* generates unnatural postures especially for the red boxer and all experts are activated equally all the time so the trajectory of the expert weights (top-row) looks like a point, which implies that the learned experts have deviated far from the original experts. On the other hand, *Enc-only* generates similar postures included in the input motions, however, the boxers can only stand in place, which implies that high-level tactics were not learned successfully.

Enc-only and *Enc-Dec-e2e* by using the boxing environment (The bottom row of Figure 7). In *Enc-only*, the player's motion is similar to that of the input motions, however, the player is not able to hit the opponent accurately and falls down after just a small amount of perturbation from the opponent. In *Enc-Dec-e2e*, the player can collect the match reward successfully during the match, however, the generated motions look unnatural. For example, the player uses an extremely wide stance to avoid losing balance when hit hard by the opponent and leans back too much to minimize the probability of being hit. The mismatch between the generated motions and the input motions can also be confirmed quantitatively by the output of the task encoder, where all the experts are always activated unlike our method and *Enc-only* (The upper row of Figure 7). Our method shows plausible tactical behaviors while preserving the style of the input motions.

5.3.2 Policy Model Components. We also evaluate how individual components in our policy model affect the resulting motion. The two key design choices are the number of experts in the motor decoder and the autoregressive parameter α that enforces smoothness on the change of the expert weights. We tested five different boxing policies using different parameters for the two design choices, where *ExpertN/ARk* means that there are N experts and the value of α is k. We first compare the performance of the imitation policies (Figure 10a). Policies having fewer than four experts show degraded performance due to the lack of representational power. To compare the performance of the competitive policies, we measure how much damage the player can deliver to the opponent during a match. More specifically, we compute the difference of $r_{pl->op}$ when the two boxers are controlled by different control policies, and we average the values over 20 matches for each policy pair (Figure10b). For example, the value 217.4N between the two polices *Expert8/AR0.9* and *Expert4/AR0.9* means that the boxer controlled by the first policy can deliver 217.4N more damage to the other boxer on average. First, *Expert8/AR0.9* performs the best whereas *Expert2/AR0.9* performs the worst, which implies that a sufficient number of experts is essential. Another observation is that the polices with lower α (*Expert8/AR0.5* and *Expert8/AR0.0*) show worse performance when compared to *Expert8/AR0.9* although their imitation polices perform similarly. This difference is probably due to the fact that higher α would help to learn a well-conditioned latent space for the motor decoder, which allows the deep RL algorithm to discover pre-trained skills more easily in the transfer learning.



(a) Closeness (b) Facing (c) Energy

Fig. 8. Ablation studies on rewards.

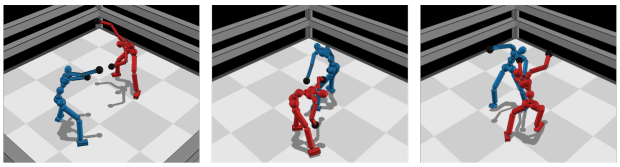
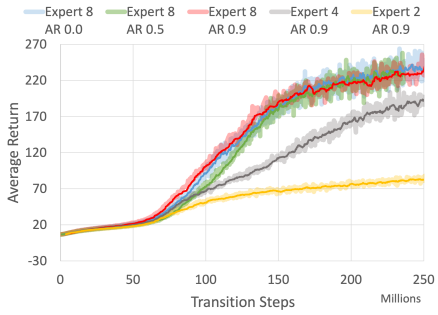
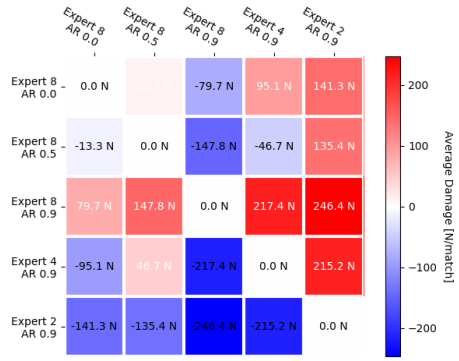


Fig. 9. A new motion style. The imitation motor decoder was trained with motions mimicking gorillas.

5.3.3 Rewards. To understand how auxiliary reward terms affect the results, we performed ablation studies for the three auxiliary terms, r_{close}, r_{facing}, and r_{energy} for the boxing environment (Figure 8). If the closeness term is ignored, boxers have only a sparse signal on physical interaction, which made competitive policies converge to sub-optimal behaviors. The boxers had difficulty returning to close positions after being apart from each other. When the facing term was discarded, the boxers occasionally attempted to punch their opponents from behind, which does not occur in real boxing matches. Finally, the control policies showed more aggressive behaviors when they were learned without the energy term. The boxers threw many punches indiscriminately regardless of the distance between the boxers. Although the control policies looked less energy efficient, we observed that the results were interesting



(a) Performance of imitation policies



(b) Performance of competitive policies. We measure the difference of damages between the two players during a match. Please note that the values are anti-symmetric.

Fig. 10. Performance comparison according to design choices

and visually appealing perhaps because aggressive players are loved by spectators more even if their tactics are ineffective.

5.3.4 Different Motion Styles. In our framework, the style of input motions are embedded into the motor decoder of the imitation policy, which eventually determines the style of the output motions generated by the competitive policies. To test how our framework adapts to a new motion style, we trained a new motor decoder with motions that are unrelated to the boxing environment, then we learned new competitive policies based on the new motor decoder. We used motions where an actor mimics gorilla's behaviors, which include a few steps and swinging arms. Figure 9 shows that the new motion style was successfully embedded into the competitive policies. One interesting observation is that the new competitive policies seemed inefficient when compared to the policies learned with the boxing motions. We suspect that the input motions having less relevance to the competitive environment (i.e. the boxing environment was originally designed for the boxing motions) might affect the final performance of the policies.

6 CONCLUSION

In this paper, we explore techniques to create control policies for physically simulated humanoids performing two-player competitive games. We propose a two-step approach that first learns an imitation policy and then transfers it into competitive policies. Boxing and fencing are demonstrated as examples. Although this approach generates competitive matches, there are limitations of our approach and promising avenues for future work, which we discuss below.

Our system requires a fair amount of computation to generate plausible competitive policies. As the number of agents involved in the environment increases, possible interactions increase exponentially, so the number of tuples required increases in a similar manner. Not only our results but also other previous studies reported that almost a billion training tuples are required for convergence [Bansal et al. 2018], whereas single-agent environments usually converge with a hundred million tuples [Peng et al. 2018]. To apply our framework to sports that involve many players like basketball or soccer, more sample-efficient methods would be necessary. This computational complexity might be solved by a breakthrough in the learning algorithms such as model-based RL algorithms, or collecting more data that can bootstrap interactions among the agents.

One assumption in our framework is that the individual skills of sports can be captured by a single actor, which enabled us to learn an effective pre-trained policy for the competitive environments later. However, there exist some two-player competitive sports where that assumption does not hold. For example, in wrestling, one player first needs to grab the other's bodies and utilizes contacts continuously to perform the skills (i.e. get a point). Such skills would not be captured by a single actor only. We expect that our framework might also work if two-player imitation policies for those skills are available, however, learning such imitation policies still remains an open problem.

Although our method is able to generate emergent behaviors for competitive policies, the naturalness of the generated motions still depends on the quality of the input reference motions. For example in boxing, professional athletes show extremely agile behaviors during the match, whereas our simulated athletes move slowly in comparison. We believe that the primary reason for this discrepancy is that the input motions used in our experiments were captured from a boxer with very limited training, who attempted to mimic professional boxers. A few studies have shown production-quality results by using an hour of coordinated motion capture data [Holden et al. 2017; Starke et al. 2019]. Using such data would be helpful to increase the naturalness and diversity of the generated motions in our framework. As the number of input motions and the included behaviors become larger and more diverse, learning an imitation policy in an end-to-end manner would fail. A hierarchical approach [Won et al. 2020] could be a remedy, where we first cluster the input motions into several categories, learn imitation policies for each category, and finally combine them into a single imitation policy.

To the best of our knowledge, the method described in this paper is the first to automatically generate human-like control policies for many degree-of-freedom, physically simulated humanoids performing two-player competitive sports. We hope that this initial work will lead to follow-up studies that improve performance and overcome the limitations to the current method. In the long term, this research direction aims to provide an immersive way that human users will be able to compete/interact by controlling intelligent

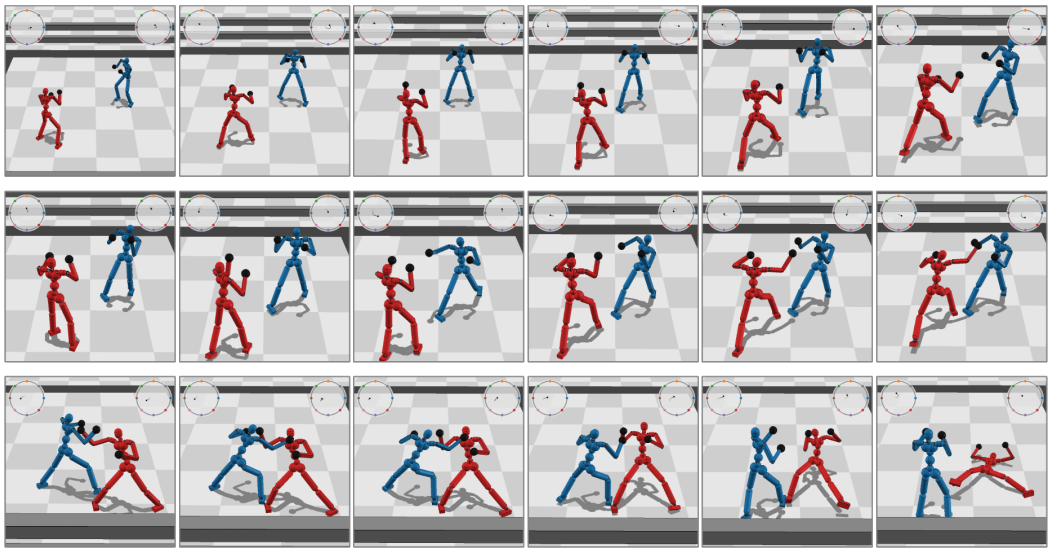


Fig. 11. An example of a fight between two boxers. The boxers move around and get closer (top-row), keep each other in check by using light jabs (middle-row), and the red boxer is knocked down by the blue boxer (bottom-row).

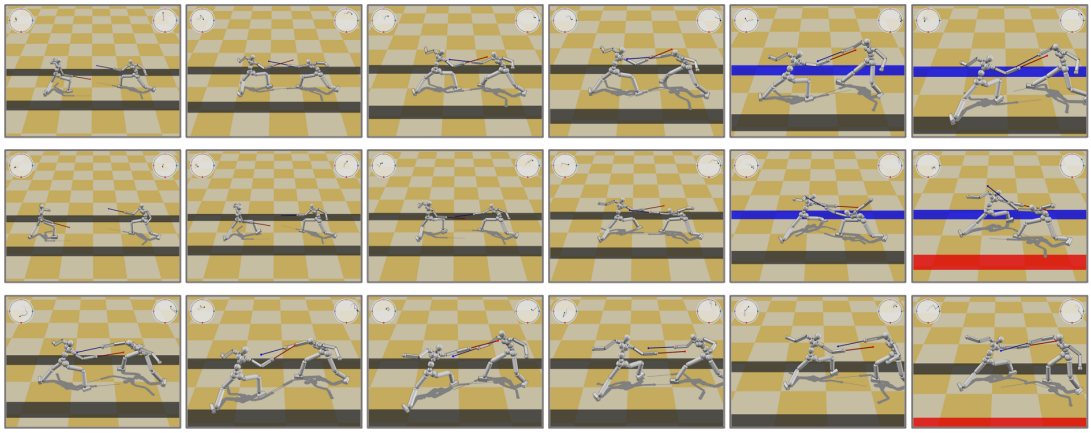


Fig. 12. Three fencing match examples. The blue fencer wins (top-row), a draw (middle-row), and the red fencer wins (bottom-row).

characters in a physically plausible manner in various applications such as video games and virtual reality.

REFERENCES

[n.d.].

Yu Bai and Chi Jin. 2020. Provable Self-Play Algorithms for Competitive Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, Vol. 119. PMLR, 551–560. http://proceedings.mlr.press/v119/bai20a.html

Bowen Baker, Ingmar Kanitscheider, Todor M. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2019. Emergent Tool Use From Multi-Agent Autocurricula. *CoRR* (2019). arXiv:1909.07528

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. 2018. Emergent Complexity via Multi-Agent Competition. arXiv:1710.03748

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: Data-driven Responsive Control of Physics-based Characters. *ACM Trans. Graph.* 38, 6, Article 206 (2019). http://doi.acm.org/10.1145/3355089.3356536

Glen Berseth, Cheng Xie, Paul Cernek, and Michiel van de Panne. 2018. Progressive Reinforcement Learning with Distillation for Multi-Skilled Motion Control. *CoRR* abs/1802.04765 (2018).

L. Busoniu, R. Babuska, and B. De Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172. https://doi.org/10.1109/TSMCC.2007.913919

Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. 2018. Physics-based motion capture imitation with deep reinforcement learning. In *Motion, Interaction and Games, MIG 2018*. ACM, 1:1–1:10. https://doi.org/10.1145/3274247.3274506

Caroline Claus and Craig Boutilier. 1998. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence (AAAI '98/IAAI '98)*. 746–752.

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 Trans. Graph.* 37, 6, Article 179 (2018). http://doi.acm.org/10.1145/3272127.3275048

CMU. 2002. CMU Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu/.

Brandon Haworth, Glen Berseth, Seonghyeon Moon, Petros Faloutsos, and Mubbasir Kapadia. 2020. Deep Integration of Physical Humanoid Control and Crowd Navigation. In *Motion, Interaction and Games (MIG '20)*. Article 15. https://doi.org/10.1145/3424636.3426894

Joseph Henry, Hubert P. H. Shum, and Taku Komura. 2014. Interactive Formation Control in Complex Environments. *IEEE Transactions on Visualization and Computer Graphics* 20, 2 (2014), 211–222. https://doi.org/10.1109/TVCG.2013.116

Edmond S. L. Ho, Taku Komura, and Chiew-Lan Tai. 2010. Spatial Relationship Preserving Character Motion Adaptation. *ACM Trans. Graph.* 29, 4, Article 33 (2010). https://doi.org/10.1145/1778765.1778770

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned Neural Networks for Character Control. *ACM Trans. Graph.* 36, 4, Article 42 (2017). http://doi.acm.org/10.1145/3072959.3073663

Junling Hu and Michael P. Wellman. 1998. Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)*. 242–250.

K. Hyun, M. Kim, Y. Hwang, and J. Lee. 2013. Tiling Motion Patches. *IEEE Transactions on Visualization and Computer Graphics* 19, 11 (2013), 1923–1934. https://doi.org/10.1109/TVCG.2013.80

Jongmin Kim, Yeongho Seol, Taesoo Kwon, and Jehee Lee. 2014. Interactive Manipulation of Large-Scale Crowd Animation. *ACM Trans. Graph.* 33, 4, Article 83 (2014). https://doi.org/10.1145/2601097.2601170

Manmyung Kim, Kyunglyul Hyun, Jongmin Kim, and Jehee Lee. 2009. Synchronized Multi-Character Motion Editing. *ACM Trans. Graph.* 28, 3, Article 79 (2009). https://doi.org/10.1145/1531326.1531385

T. Kwon, Y. Cho, S. I. Park, and S. Y. Shin. 2008. Two-Character Motion Analysis and Synthesis. *IEEE Transactions on Visualization and Computer Graphics* 14, 3 (2008), 707–720. https://doi.org/10.1109/TVCG.2008.22

Taesoo Kwon, Kang Hoon Lee, Jehee Lee, and Shigeo Takahashi. 2008. Group Motion Editing. *ACM Trans. Graph.* 27, 3 (2008), 1–8. https://doi.org/10.1145/1360612.1360679

Jehee Lee and Kang Hoon Lee. 2004. Precomputing Avatar Behavior from Human Motion Data. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '04)*. 79–87. https://doi.org/10.1145/1028523.1028535

Kang Hoon Lee, Myung Geol Choi, and Jehee Lee. 2006. Motion Patches: Building Blocks for Virtual Environments Annotated with Motion Data. *ACM Trans. Graph.* 25, 3 (2006), 898–906. https://doi.org/10.1145/1141911.1141972

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

Eric Liang, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. arXiv:1712.09381

Michael L. Littman. 1994. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning (ICML'94)*. 157–163.

C. Karen Liu, Aaron Hertzmann, and Zoran Popović. 2006. Composition of Complex Optimal Multi-Character Motions. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*. 215–222.

Libin Liu and Jessica Hodgins. 2017. Learning to Schedule Control Fragments for Physics-Based Characters Using Deep Q-Learning. *ACM Trans. Graph.* 36, 3, Article 42a (2017). http://doi.acm.org/10.1145/3083723

Libin Liu and Jessica Hodgins. 2018. Learning Basketball Dribbling Skills Using Trajectory Optimization and Deep Reinforcement Learning. *ACM Trans. Graph.* 37, 4, Article 142 (2018). http://doi.acm.org/10.1145/3197517.3201315

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 6382–6393.

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

T. T. Nguyen, N. D. Nguyen, and S. Nahavandi. 2020. Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Transactions on Cybernetics* 50, 9 (2020), 3826–3839. https://doi.org/10.1109/TCYB.2020.2977374

Afshin OroojlooyJadid and Davood Hajinezhad. 2020. A Review of Cooperative Multi-Agent Deep Reinforcement Learning. arXiv:1908.03963

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. 8024–8035.

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 Trans. Graph.* 37, 4, Article 143 (2018). http://doi.acm.org/10.1145/3197517.3201311

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

Yevgeny Seldin and Aleksandrs Slivkins. 2014. One Practical Algorithm for Both Stochastic and Adversarial Bandits. In *Proceedings of the 31st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 32)*. 1287–1295. http://proceedings.mlr.press/v32/seldinb14.html

Hubert P. H. Shum, Taku Komura, Masashi Shiraishi, and Shuntaro Yamazaki. 2008. Interaction Patches for Multi-Character Animation. *ACM Trans. Graph.* 27, 5 (2008). https://doi.org/10.1145/1409060.1409067

Hubert P. H. Shum, Taku Komura, and Shuntaro Yamazaki. [n.d.]. Simulating Interactions of Avatars in High Dimensional State Space. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games (I3D '08)*. 131–138. https://doi.org/10.1145/1342250.1342271

Hubert P. H. Shum, Taku Komura, and Shuntaro Yamazaki. 2007. Simulating Competitive Interactions Using Singly Captured Motions. In *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology (VRST '07)*. 65–72. https://doi.org/10.1145/1315184.1315194

H. P. H. Shum, T. Komura, and S. Yamazaki. 2012. Simulating Multiple Character Interactions with Collaborative and Adversarial Goals. *IEEE Transactions on Visualization and Computer Graphics* 18, 5 (2012), 741–752. https://doi.org/10.1109/TVCG.2010.257

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural state machine for character-scene interactions. *ACM Trans. Graph.* 38, 6 (2019), 209:1–209:14. https://doi.org/10.1145/3355089.3356505

Jie Tan, C. Karen Liu, and Greg Turk. 2011. Stable Proportional-Derivative Controllers. *IEEE Computer Graphics and Applications* 31, 4 (2011), 34–44. https://doi.org/10.1109/MCG.2011.30

Kevin Wampler, Erik Andersen, Evan Herbst, Yongjoon Lee, and Zoran Popović. 2010. Character Animation in Two-Player Adversarial Games. *ACM Trans. Graph.* 29, 3, Article 26 (2010). https://doi.org/10.1145/1805964.1805970

Ziyu Wang, Josh Merel, Scott Reed, Greg Wayne, Nando de Freitas, and Nicolas Heess. 2017. Robust Imitation of Diverse Behaviors. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. http://dl.acm.org/citation.cfm?id=3295222.3295284

Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. 2020. DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

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

Jungdam Won and Jehee Lee. 2019. Learning Body Shape Variation in Physics-based Characters. *ACM Trans. Graph.* 38, 6, Article 207 (2019). http://doi.acm.org/10.1145/3355089.3356499

Jungdam Won, Kyungho Lee, Carol O'Sullivan, Jessica K. Hodgins, and Jehee Lee. 2014. Generating and Ranking Diverse Multi-Character Interactions. *ACM Trans. Graph.* 33, 6, Article 219 (2014). https://doi.org/10.1145/2661229.2661271

Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. 2020. ALLSTEPS: Curriculum-driven Learning of Stepping Stone Skills. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

Barbara Yersin, Jonathan Maïm, Julien Pettré, and Daniel Thalmann. 2009. Crowd Patches: Populating Large-Scale Virtual Environments for Real-Time Applications. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games (I3D '09)*. 207–214. https://doi.org/10.1145/1507149.1507184

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

Victor Brian Zordan and Jessica K. Hodgins. 2002. Motion Capture-Driven Simulations That Hit and React. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '02)*. 89–96. https://doi.org/10.1145/545261.545276

Victor Brian Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. 2005. Dynamic Response for Motion Capture Animation. *ACM Trans. Graph.* 24, 3 (2005), 697–701. https://doi.org/10.1145/1073204.1073249