Strategy and Skill Learning for Physics-based Table Tennis Animation

Jiashun Wang
jiashunw@cmu.edu
Carnegie Mellon University
USA

Jessica Hodgins
jkh@cmu.edu
Carnegie Mellon University
and The AI Institute
USA

Jungdam Won
jungdam@imo.snu.ac.kr
Seoul National University
South Korea

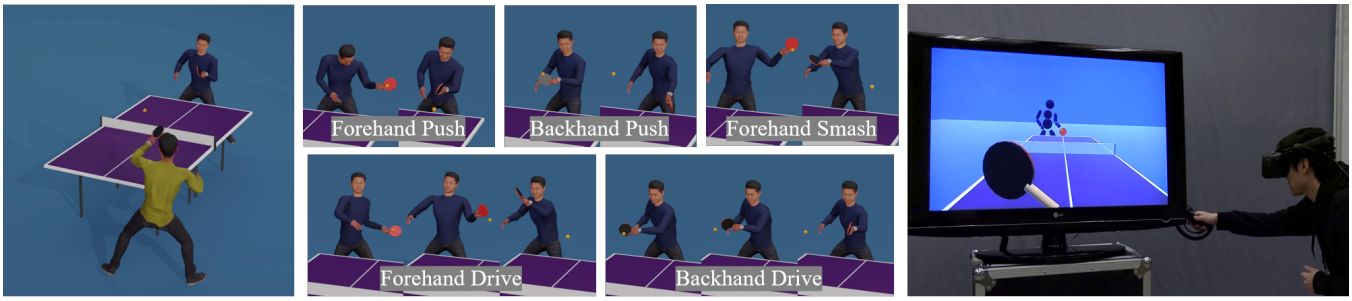

Figure 1: On the left, two physically-simulated agents engage in a competitive match, controlled by our strategy and skill controllers. The center panels show the five learned skills: Forehand Drive, Push and Smash, Backhand Drive and Push, showcasing the skill controller's ability to execute a diverse set of skills. On the right, a human interacts with the simulated agent through VR.

ABSTRACT

Recent advancements in physics-based character animation leverage deep learning to generate agile and natural motion, enabling characters to execute movements such as backflips, boxing, and tennis. However, reproducing the selection and use of diverse motor skills in dynamic environments to solve complex tasks, as humans do, still remains a challenge. We present a strategy and skill learning approach for physics-based table tennis animation. Our method addresses the issue of mode collapse, where the characters do not fully utilize the motor skills they need to perform to execute complex tasks. More specifically, we demonstrate a hierarchical control system for diversified skill learning and a strategy learning framework for effective decision-making. We showcase the efficacy of our method through comparative analysis with state-of-the-art methods, demonstrating its capabilities in executing various skills for table tennis. Our strategy learning framework is validated through both agent-agent interaction and human-agent interaction in Virtual Reality, handling both competitive and cooperative tasks.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**; Motion Processing.

KEYWORDS

Character Animation, Physics-based Characters, Deep Reinforcement Learning, Multi-character Interaction, Virtual Reality, Table Tennis

ACM Reference Format:
Jiashun Wang, Jessica Hodgins, and Jungdam Won. 2024. Strategy and Skill Learning for Physics-based Table Tennis Animation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657437

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07.
https://doi.org/10.1145/3641519.3657437

1 INTRODUCTION

The integration of deep learning into physics-based character animation has led to significant advancements in generating agile and natural motion, enhancing the lifelike quality of characters in complex environments. To increase the versatility of these characters, it is essential to ensure that their skills can be reused in environments or conditions that may not precisely match their training data. To achieve this goal, recent approaches have focused on learning reusable skill embeddings. These approaches are typically trained in two stages. Initially, characters learn various skill embeddings by imitating reference motions. Then, in the task training stage, they apply these skills to accomplish diverse tasks. These approaches have demonstrated remarkable success in generating natural motion in various environments.

However, when the differences between the skills are subtle, these approaches often suffer from mode collapse during the task training phase. Specifically, although agents (a.k.a. characters) can learn various skills during the imitation stage, they tend to use

a limited set of skills for the downstream tasks, neglecting the diversity of their learned skills in the imitation stage. Thus, mode collapse restricts the agents' potential in scenarios that require a diverse set of skills. Mode collapse also restricts exploration during RL training, resulting in sub-optimal task performance.

Another relatively unexplored topic relates to the decision strategy of agents, particularly their ability to dynamically formulate decision strategies that encompass skill selection and associated skill goals in response to task demands. Most previous studies either have not required a diverse skill set or have relied on a human user to manually determine skills for the agents. Agents have generally not been equipped with the capability to employ different strategies to adapt to complex and dynamic environments.

Our research introduces a learning approach to enhance both the skill and strategic decision-making capabilities of physically simulated agents. First, we develop a hierarchical skill controller that enables agents to utilize different table tennis skills and transition among them rapidly. This controller effectively addresses mode collapse during task training. Second, we develop a method for strategy learning, enabling agents to explicitly select and utilize specific skills for different types of interaction, whether competitive or cooperative. An overview of the results is in Figure 1.

We demonstrate the effectiveness of our approach through two interaction environments: a table tennis match played between two simulated agents and a match between a human and a simulated agent in virtual reality (VR). In the agent-agent environment, the agents demonstrate improved skill diversity and decision strategy in simulated table tennis matches compared to results predicted by the previous techniques. In the human-agent interaction environment, we evaluate both cooperative and competitive scenarios in real-time interactions between humans and agents. These environments not only validate our approach but also provide platforms for future research into complex agent behaviors and human-agent dynamics. Code and data for this paper are at https://jiashunwang.github.io/PhysicsPingPong/.

We summarize the contributions of this paper as follows:

- A hierarchical skill controller that empowers physically simulated agents to explicitly perform various skills, enabling rapid skill transitions. An interaction learning framework designed to create a decision strategy allows agents to continually learn and adapt, meeting the demands of competition or cooperation in dynamic environments with other agents and with humans.
- Novel results demonstrating our learning framework's capacity to generate intelligent decisions and natural motions for table tennis in two scenarios: agent-agent interactions in a simulated environment and human-agent interactions in a VR environment. The agent-agent environment is a platform for developing and testing competitive and cooperative algorithms while the VR environment allows natural human-agent interactions.

2 RELATED WORK

We review the closest related work in physics-based character animation with reusable skills and multi-character animation. We review studies on transitions among skills as we develop a method for skill selection and transition. We further discuss relevant research in human-agent interaction in VR.

2.1 Physics-based Character Animation

Incorporating physical laws into character animation allows for the development of controllers that generate more realistic behaviors [Hodgins et al. 1995; Laszlo et al. 1996]. Optimization techniques, such as trajectory optimization [de Lasa et al. 2010; Mordatch et al. 2012; Yin et al. 2008] and sampling-based methods [Liu et al. 2016, 2010] have been widely explored. Recently, deep reinforcement learning (DRL) has been shown to substantially enhance control capabilities [Liu and Hodgins 2017; Peng et al. 2017]. Due to its flexibility and ease of use, DRL methods eliminate the need for designing complex objective functions while delivering outstanding results and have attracted significant research interest as a result.

Data-driven methods have become prevalent in physics-based character animation studies since a DRL-based method was introduced by Peng et al. [2018]. The idea has been extended for handling larger datasets [Bergamin et al. 2019; Won et al. 2020] and for allowing recombination of existing state transitions [Peng et al. 2021]. Recently, much attention has been paid to reusable motor skills. The idea is to learn a latent space of reference motions and then to reuse the learnt space for downstream tasks. Various latent models have been studied such as encoder-decoders with autoregression [Merel et al. 2019; Won et al. 2021], spherical embedding [Dou et al. 2023; Peng et al. 2022; Tessler et al. 2023], conditional variational autoencoder (VAE) [Won et al. 2022; Yao et al. 2022], and vector-quantized VAE [Zhu et al. 2023]. Some researchers have also proposed partwise models to maximize reusability of reference motions [Bae et al. 2023; Xu et al. 2023].

Our system is designed for table tennis games, involving two players (i.e., agents). Two or more agents have been created primarily with kinematic approaches [Kwon et al. 2008; Liu et al. 2006; Shum et al. 2008, 2012; Wampler et al. 2010]. There exist two recent approaches [Won et al. 2021; Zhu et al. 2023] demonstrating examples of physically simulated boxing. Zhang et al. [2023] build a system to learn tennis skills from broadcast videos and produce rallies with a mirrored opponent. In their approach, kinematics-based motion generation is utilized first, followed by physics-based tracking, relying on residual forces and extra arm control for successful strikes. Skill and target selection are not learned but rather performed manually or randomly to create a scene including two players. In contrast, our method learns not only agile and precise motor control to strike the ball but also strategies to select skills and targets based on the movement of the opponent and the ball.

2.2 Transition of skills

Option-based methods [Bagaria and Konidaris 2020; Jain et al. 2021; Klissarov et al. 2017; Konidaris and Barto 2009; Sutton et al. 1999] represent skills as *options*, which are sequentially constructed, with each option's execution in the chain enabling the agent to execute the subsequent option. Lee et al. [2019] propose learning additional transition policies to connect primitive skills and introduce proximity predictors, which yield rewards based on proximity suitable for initial states for the next skill. One challenge of transitioning between different skills to chain long-horizon tasks is addressed

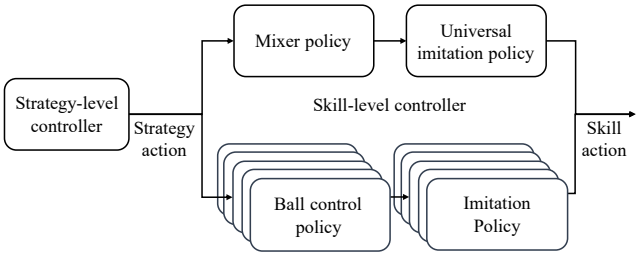

Figure 2: An overview of our method. Strategy action includes the skill command and ball's target landing location. Skill action includes the target joint angles for PD controllers, blended from the outputs of imitation policies.

by terminal state regularization [Lee et al. 2021]. Behavior Trees are also a common method for planning the transition between different states [Cheng et al. 2023; French et al. 2019; Marzinotto et al. 2014]. These methods achieve skill transitions by ensuring that the terminal state of the previous stage is close to the initial state of the next stage. While these methods work well for tasks that are not time-sensitive, table tennis, which involves high-speed movements and rapid responses, poses a challenge as players do not always hit the ball from a well-defined initial state.

2.3 Human-agent interaction

Research has focused on human sports training within VR [Liu et al. 2020; Pastel et al. 2023]. However, these studies often lack a physically simulated opponent. There are commercial games that allow people to interact with an agent in VR for sports activities, such as boxing, golf, and badminton. Eleven Table Tennis [2016] is a VR-based table tennis game similar to the one we have constructed, which enables a human to play with an agent. However, this agent is not simulated with full-body dynamics, rather it is simulated with only a floating head and a floating paddle. Advances in GPU-accelerated simulation and our control algorithm, enable us to create a physically-simulated agent with full-body dynamics that can play in real-time with humans. Another relevant area involves enhancing the agent's capabilities with *human-in-the-loop* methodologies [Brenneis et al. 2021; Li et al. 2022; Seo et al. 2023; Wang et al. 2023] using extended reality. Our work differs from previous studies by bringing humans and agents into a unified environment allowing bidirectional physical interaction, where they can cooperate and compete.

3 METHOD OVERVIEW

We propose a hierarchical approach that includes a strategy-level controller and a skill-level controller. The strategy-level controller takes the states of the agent, opponent, and ball as inputs, and outputs a strategy action, which includes the skill to use and the target landing location for the ball. Meanwhile, the skill-level controller takes the states of the agent and ball, along with the strategy action as inputs, and then generates a skill action, which includes the target joint angles for PD controllers. An overview of our method is in Figure 2 and Figure 3 shows the architecture of our method.

4 SKILL-LEVEL CONTROLLER

Three stages are required to train our skill-level controller. Initially, we train imitation policies using the motion capture data. Then the ball control policy for each skill is learned, which enables the agent to hit back balls using the corresponding imitation policy. Finally, we learn a policy that enables the agent to perform various skills sequentially while making plausible transitions among them. We call this policy the mixer policy. Once the skill-level controller is trained, the agent can proficiently and continuously execute various skills, sending balls to diverse target locations.

4.1 Imitation Policy

We first categorize the motion capture dataset into five subsets corresponding to each skill. This subdivision allows us to train the skill-specific imitation policies. We also utilize all the data to train a universal imitation policy. The imitation policy is represented as $\pi^i(a^i|s, z^i)$, where $i \in \{1, 2, 3, 4, 5, u\}$, $1 \sim 5$ are indices of different skills and u is the index of the universal imitation policy. z^i is a latent variable sampled from a hyper-sphere distribution, and s is the agent's state. The goal of the imitation policy is to output an action a^i that leads to simulated motions similar to the reference motions. Thus, each skill-specific imitation policy generates motions similar to its corresponding reference motion in each skill subset, while the universal imitation policy generates motions encompassing the entire motion capture dataset. When solving specific tasks in the later stage, using a single universal imitation policy trained with a variety of motions often leads to the mode collapse problem. The agent does not explore various available skills enough; instead, it repeats very limited skills, and the task performance remains suboptimal. Our controller design is inspired by mixture-of-experts and mitigates this problem. Each imitation policy $\pi^i(a^i|s, z^i)$ is built by the adversarial framework ASE [Peng et al. 2022], where the policy is updated so that it tricks a motion discriminator D^i. The transitions $d_{M^i}(s, s')$ existing in the motion capture dataset are used as positive samples while the transitions $d_{\pi^i}(s, s')$ generated from the policy π^i are used as negative samples. The discriminator is trained by minimizing:

$$\begin{aligned}\min_{D^i} & -\mathbb{E}_{d_{M^i}(s,s')}\log(D^i(s,s')) - \mathbb{E}_{d_{\pi^i}(s,s')}\log(1-D^i(s,s')) \\ & + \lambda_{gp}\mathbb{E}_{d_{M^i}(s,s')}\left\|\nabla_\phi D^i(\phi)\big|_{\phi=(s,s')}\right\|^2, \end{aligned} \tag{1}$$

where the last term is a gradient penalty regularization with a constant factor λ_{gp}. We train encoders q^i to encourage correspondence between the transition (s, s') and the latent variable z^i. The encoder is modeled as a von Mises-Fisher distribution and it is trained by maximizing its log-likelihood:

$$\begin{aligned}\max_{q^i} & \mathbb{E}_{p(z^i)}\mathbb{E}_{d^{\pi^i}(s,s'|z^i)}[\log q^i(z^i|s,s')], \\ & q^i(z^i|s,s') = \frac{1}{Z}exp(\mu_{q^i}(s,s')^T z^i)\end{aligned} \tag{2}$$

where $\mu_{q^i}(s, s')$ is the mean of the distribution, and Z is a normalization constant. Given a discriminator D^i, the reward to train π^i is defined as:

$$r_t = -\log(1 - D^i(s_t, s_{t+1})) + \beta \log q^i(z_t^i|s_t, s_{t+1}). \tag{3}$$

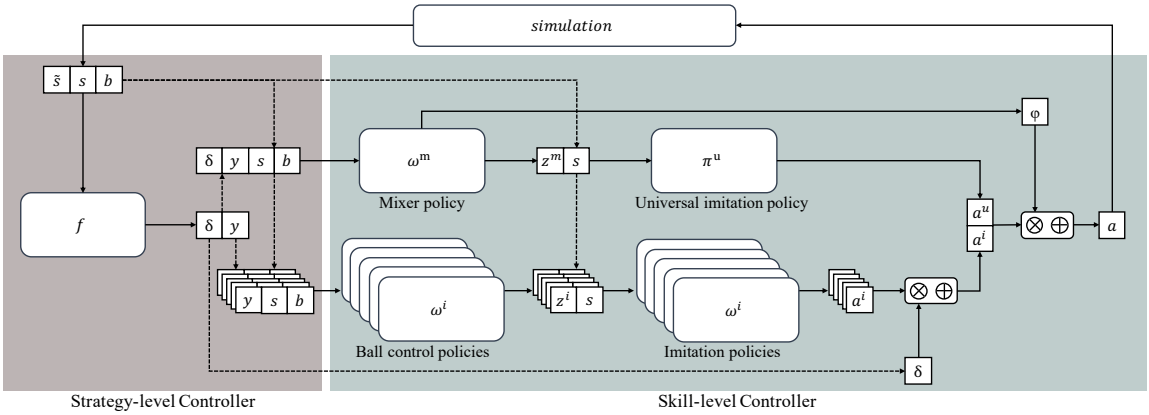

Figure 3: The architecture of our method. We train the skill-level controller through the stages of imitation policies, ball control policies, and finally, the mixer policy. We train the strategy-level controller after the skill-level controller is ready and its weight is frozen. ⊗⊕ stands for the weighted sum in Equation 8.

where β is the relative weight. Additionally, a diversity term is included to encourage different latent variables to represent distinct motions. Bringing everything together, the objective of π^i becomes:

$$\max_{\pi^i} \mathbb{E}_{p(Z)}\mathbb{E}_{p(\tau|\pi^i,Z)}[\sum_{t=0}^{T}\gamma^t(r_t)] \tag{4}$$
$$-\lambda_{D^i}\mathbb{E}_{d^{\pi^i}(s)}\mathbb{E}_{z_1^i,z_2^i\sim p(z^i)}\left[\left(\frac{D_{KL}(\pi^i(\cdot|s,z_1^i),\pi^i(\cdot|s,z_2^i))}{0.5(1-z_1^iz_2^i)}-1\right)^2\right],$$

where $D_{KL}(\cdot||\cdot)$ measures the KL-divergence between two distributions, z_1^i and z_2^i refer two different latent variables, γ is the discount factor and λ is a constant used to balance the weight.

4.2 Ball Control Policy

Once the agent can imitate each skill $i \in \{1,2,3,4,5\}$, we train ball control policies $\omega^i(z^i|s,b,y)$ to enable the agent to hit and move a ball launched from a random location to the desired location, where s denotes the state of the agent, b represents the state of the ball and y is the target landing location for the ball. The task reward r is a composite of three terms: the paddle reward r_p, the ball reward r_b, and the style reward r_s,

$$r(t) = w_p r_p(t) + w_b r_b(t) + w_s r_s(t), \tag{5}$$

where w_p, w_b, and w_s are the relative weights. The paddle reward r_p encourages the agent to position the paddle close to the ball. The reward is defined as:

$$r_p(t) = \begin{cases} \exp(-4||x_p(t)-x_b(t)||^2), & \text{if } C_{bp}(t)=0, \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where $x_p(t)$ and $x_b(t)$ represent the positions of paddle and ball, respectively, $C_{bp}(t)$ is a binary variable representing contact states. $C_{bp}(t)=0$ means the ball has not contacted the paddle until time t, while $C_{bp}(t)=1$ indicates the ball has contacted the paddle at time t or contacted previously before time t. It will be reset to 0 whenever the next ball is launched. The ball reward r_b is given by:

$$r_b(t) = \begin{cases} 1+\exp(-4||x_c(t)-x_t(t)||^2), \\ \qquad \text{if } C_{bp}(t)=1 \text{ and } C_{bt}(t)=0, \\ 0, \text{ otherwise,} \end{cases} \tag{7}$$

where $x_t(t)$ is the target landing location of the ball, $x_c(t)$ represents the anticipated landing location on the table, calculated using Newton's equation of motion for the point mass (i.e., quadratic trajectory), with its state corresponding to the current position and velocity of the ball. $C_{bt}(t)$ is a binary variable checking the contact history between the ball and the table, which is updated similarly to $C_{bp}(t)$. The agent receives the maximum reward when it successfully hits the ball and the ball moves toward the target location. We also apply the style reward, $r_s = -\log(1-D^i(s_t,s_{t+1}))$ in the task training similarly to ASE [Peng et al. 2022], where D^i is the discriminator learned during the previous stage.

4.3 Mixer Policy

While our agent can play table tennis using the ball control policies with the corresponding imitation policies, its capability is limited to repeating a single skill. Simply transitioning from one controller to another during play often leads to failure due to a mismatch between the end state of one skill to the start state of the next. To create plausible transitions among the different skills, we learn a mixer policy $\omega^m(z^m|s,b,\delta,y)$, which takes the agent state s, the ball state b, and the strategy action (δ,y) as input, where δ is a one-hot vector determining the skill to use and y is the target ball landing location, then generates the latent variable for the universal imitation policy π^u and a set of blending weights φ, mixing the skill actions in a joint-wise manner. In other words, φ determines which policy the agent relies on among the transition and five different skills. The target joint angles for PD controllers are computed as

$$a = \varphi \odot \pi^u(\cdot|s,z^u) + (1-\varphi)\odot\sum_{i=1}^{5}\delta_i\pi^i(\cdot|s,z^i) \tag{8}$$

ALGORITHM 1: Strategy learning

Input: Number of iterations N, interaction environment Env.
Output: Updated policy f.
$f \leftarrow$ Random initialization ;
for $i \leftarrow 1$ **to** N **do**
  $\{(o_k^{\text{expert}}, c_k^{\text{expert}})\}_{k=1}^{K} \leftarrow \text{Interact}(Env, f)$;
  Apply stochastic gradient descent to update f using Equation 9
end

where $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$ is a one-hot vector indicating the skill selected. While training the mixer policy, the agent is asked to perform the ball control task with randomly launched balls, randomly selected skills, and random target locations. The same rewards used for learning ball control policies are employed, and the weights of all other policies remain frozen.

5 STRATEGY-LEVEL CONTROLLER

The strategy-level controller is developed by iterative behavior cloning inspired by [Oh et al. 2018]. More specifically, we first collect interaction data by randomly sampling strategy actions during agent-agent play or human-agent interactions with VR. This data is then used to update the strategy-level controller, and we repeat this process by collecting new interaction data with the latest strategy-level controller. When collecting interaction data, there are two options: competition and cooperation. To train a competitive strategy, we choose data that results in victories, whereas in a cooperative strategy, we choose sequences where the opponent successfully catches the ball.

A strategy-level controller produces a skill index and a target landing location repeatedly so that they satisfy the requirements of different applications. More specifically, the strategy-level controller f takes the strategy observation $o = (s, \tilde{s}, b)$ as input where s, $\tilde{s}$, and b are the agent state, the opponent state, and the ball state, respectively, then outputs the strategy action $c = (\delta, y)$, where δ is a one-hot vector determining the skill to use, and y is the target landing location of the ball. The strategy action is updated when the ball starts moving from the opponent to the agent. To effectively learn a strategy-level controller, we adopt a behavior cloning approach with iterative refinement, aiming to learn strategies from available expert demonstrations $\{(o_k^{\text{expert}}, c_k^{\text{expert}})\}_{k=1}^{K}$ (see Algorithm 1). As a structure of the controller, we utilize a Conditional Variational Autoencoder (CVAE) to model the stochastic nature inherent in sports gameplay. During training, the CVAE encoder takes o and c as inputs and generates the mean μ and variance σ^2 of the posterior Gaussian distribution $Q(u|\mu, \sigma^2)$. We then sample a latent variable u from this distribution and concatenate it with observation o as input for the decoder, which reconstructs the action c'. The training loss is defined as:

$$\sum_{k=1}^{K} ||c_k^{\text{expert}} - c_k'|| + \beta_{KL} D_{KL}(Q(u|\mu_k, \sigma_k^2)||\mathcal{N}(0, I)), \qquad (9)$$

where $D_{KL}(\cdot||\cdot)$ measures the KL divergence between the two distributions and β_{KL} is the relative weight. During inference the decoder is utilized solely, it takes a randomly sampled latent variable u and the observation o, and then generates the strategy action that guides the agent to perform a corresponding skill. If the opponent successfully returns the ball, this process repeats. We collect expert demonstrations from two different interaction environments (Env in Algorithm 1). The details of each environment will be explained in Section 6.

6 INTERACTION ENVIRONMENT

We introduce the agent-agent and human-agent interaction environments that we build to validate the strategy learning approach.

The agent-agent interaction environment is an environment where two virtual agents play table tennis with each other (Figure 1 left column). We name one agent as *our agent* and the other as *the opponent*. In the process of learning a strategy-level controller for our agent, the opponent uses a fixed heuristic strategy-level controller while the controller for our agent is updated iteratively. More specifically, we let our agent the opponent play with each other using their own strategy-level controllers, collect those demonstrations, and then use them to update our agent's controller. If our goal is to learn a competitive strategy, that can beat the opponent, we selectively use demonstrations leading to wins. On the other hand, we use demonstrations where the opponent successfully returns the ball when aiming to learn a cooperative strategy. In our system, we utilize two types of heuristic strategy-level controllers: a random strategy and a video strategy. The random strategy selects skills and target landing locations randomly from a uniform distribution. The video strategy is constructed by using broadcast videos. We extract expert demonstrations $\{(o_k^{\text{video}}, c_k^{\text{video}})\}_{k=1}^{K}$ from existing broadcast videos (20 minutes in total). Subsequently, we train a CVAE using the behavior cloning method.

The human-agent interaction environment allows a human user to play with a virtual agent. In our system, the user interacts with an agent by using a VR device, including a head-mount display and a hand controller ((Figure 1 right column)). The VR interface operates through Unity while the physics-based simulation runs on Isaac Gym [Makoviychuk et al. 2021]. To enable the simulated agent to interact with a human user, we physically simulate the user's paddle, with its position and orientation controlled via signals from the VR interface. Specifically, for paddle control, we use the VR hand controller's Cartesian pose q_{user} and the simulated paddle pose q_{sim} to calculate the target velocity $\dot{q}_{\text{target}} = (q_{\text{user}} - q_{\text{sim}})/\Delta t$, where Δt is the simulation step. We use this target velocity as an input to the velocity controller provided by the simulator. For visualization, Unity takes the state of the simulated agent, user's paddle, and ball as inputs and renders them using visualization assets. This implementation significantly reduces the amount of information exchange compared to a previous study that sent stereo images [Seo et al. 2023], enabling real-time interaction and gameplay. By considering a human user as the opponent, the strategy-level controller of the agent can be built through the same pipeline used for the agent-agent interaction environment.

7 EXPERIMENTS

We evaluate the skill-level controller based on motion quality and task performance. We assess the strategy-level controller by examining its effectiveness in agent-agent and human-agent interaction

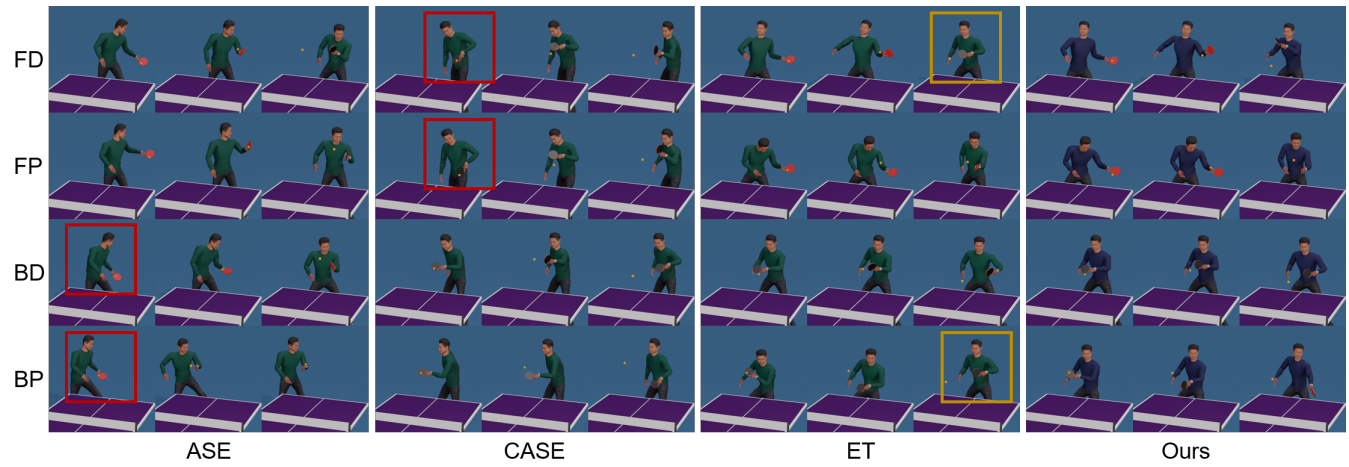

Figure 4: Comparison with other methods with four skill commands. ASE and CASE may use wrong skills as shown in the red box. ET may terminate earlier to return to a preparation pose, as shown in the yellow boxes.

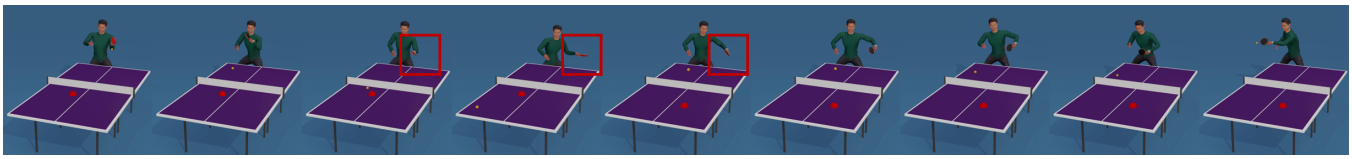

Figure 5: Transition results with only using forehand and backhand drive controllers. Both controllers are trained with random initialized configurations from the motion capture data. As shown in the red boxes, the agent attempts to use another forehand drive before the next ball is launched, which prevents it from switching back to a backhand drive in time.

environments, with the demands of both competition and cooperation scenarios.

7.1 Skill evaluation

We evaluate the skill performance from motion quality and task performance. The evaluation of motion quality measures the naturalness of generated motions when given the desired skill command and whether the agent performs the correct skill. The evaluation of task performance measures the overall proficiency in playing table tennis. We compare our method with two state-of-the-art methods, ASE [Peng et al. 2022] and CASE [Dou et al. 2023], as well as an explicit transition model (ET) which is a variant of our method with the mixer policy ω^m removed from our model. We train an explicit controller to handle skill transitions by taking over the control when the ball passes the net until it is returned to the agent. The controller is also built using the ball control-imitation architecture. The key difference between our approach and ET is that ours provides continuous action blending with the selected skill's action at every time step, whereas ET does not.

7.1.1 Motion quality. We design three metrics to evaluate the motion quality, particularly to evaluate the naturalness and mode collapse. The first metric is *Discriminator Score*, which measures how similar the current strike motion is to the reference motion of i-th target skill. Because we have five skills, we train a discriminator D^i_{test} for each skill and utilize the following equation to calculate the score:

$$\text{Discriminator Score i} = \frac{1}{T}\sum_{t=0}^{T-1} -\log(1 - D^i_{test}(s_t, s_{t+1})), \quad (10)$$

where T is the length of a single strike motion. The details of training D^i_{test} will be introduced in Appendix D. The second metric is *Skill Accuracy*, to measure whether the agent performs the correct skill given the target skill command. Specifically, given a motion sequence, we first classify it by taking the index of the discriminator which provides the highest value. Then, we compare it with the target skill command to calculate accuracy. The third metric, *Diversity Score*, is designed to test whether motions for drive and push commands are distinctive enough. In table tennis, motions within each skill category (e.g., forehand drive vs. forehand push) might exhibit subtle differences, even though their roles in gameplay can be significantly distinct. *Diversity Score* measures the capability of distinguishing motions that are visually similar. It is calculated by

$$\text{Diversity Score} = \frac{1}{2N^2}\sum_{i\in\{1,3\}}\sum_{m=1}^{N}\sum_{n=1}^{N}||s^i_m - s^{i+1}_n||, \quad (11)$$

where s^i is the state that the agent hits the ball under skill command i. Specifically $i \in \{1, 3\}$ stands for forehand drive and backhand drive and $i+1 \in \{2, 4\}$ stands for forehand push and backhand push respectively. N is the total number of hits for each skill command. We only take into account the moment when the agent's paddle makes contact with the ball to calculate this score. The first two metrics evaluate a general skill mode collapse problem, for example,

using forehand motions when being asked to use backhand. The third metric is specifically designed to measure if the agent has the ability to accurately perform drive and push skills.

The evaluation results are reported in Table 1, where the values are computed with 10k balls randomly launched toward the agent equipped with the respective skill controller. For the *Discriminator Score*, our method significantly surpasses ASE and CASE, and achieves 15.6% higher score than ET. These results prove our method generates motions that are the most similar to the reference target skill. As shown in the *Skill Accuracy* results, our method uses the correct skills to hit the ball in most cases (0.76 in Table 1). While ASE and CASE only use the correct skill with an accuracy of 0.38 and 0.47. In *Diversity Score*, our method achieves 30.7%, 32.3%, and 9.4% higher scores than ASE, CASE, and ET respectively. We also show a qualitative comparison in Figure 4. We find ASE and CASE often use forehand skills when asked to use backhand skills, or vice-versa, as shown in the red boxes in Figure 4. And we can't observe any forehand smash skill. Even when the correct skills are used, the naturalness remains insufficient. ET often does not complete the skills; instead, the skills are terminated earlier to return to a preparation pose, as shown in the yellow box in Figure 4. ASE and CASE often overlook skill commands, tending to use relatively fewer skills. This error occurs because, during the task training, these methods fall into mode collapse, making it challenging to effectively explore various skills. In contrast, our approach leverages an idea of the mixture-of-experts approach to avoid this problem. We further test the use of individual skill controllers without any design for transitions. Each skill controller is trained with randomly initialized configurations sampled from the motion capture data. As shown in Figure 5, after executing a forehand drive, the agent attempts another forehand drive before the next ball is launched—a typical behavior for single-skill controllers. This unnecessary movement prevents it from switching to a backhand drive in time, ultimately causing a missed shot.

7.1.2 Task performance. To assess the task performance of the skill controller, we evaluate two aspects: sustainability and accuracy. Sustainability is determined by the average number of successful continuous returns, while accuracy is measured by the average distance in meters between the target landing location and the actual contact location on the table. Besides testing on the training distribution, we collect some ball tracking data with faster ball trajectories from a match between high-ranking players and evaluate whether each method can perform well with the testset of the ball tracking data. We report the evaluation results in Table 2. The numbers in parentheses are the results of the fine-tuning experiments. Our method can achieve the largest number of average hits and the second-best accuracy. Although ET can achieve higher accuracy, it is not sustainable, especially for more challenging balls. It only achieves an average of 3.66 hits because it often lacks time to respond to the next ball due to the explicit transition design.

7.1.3 Blending weights of the mixer policy. We test the agent with different skills to hit the ball and visualize the average blending weights φ of the shoulder, elbow, and wrist joints in Figure 8. We can observe that the weights of the mixer policy are usually lowest at the moment the paddle contacts the ball, and higher before and after transitions between different skills. It indicates a reliance on the pre-trained ball control policy during ball strikes, and on the mixer policy during transitions.

Table 1: Comparisons on *Discriminator Score*, *Skill Accuracy*, and *Diversity Score*.

	ASE	CASE	ET	Ours
Discriminator Score	1.62	2.28	4.95	**5.72**
Skill Accuracy	0.38	0.47	0.69	**0.76**
Diversity Score	6.13	6.05	7.32	**8.01**

Table 2: Task performance evaluation. Our method can achieve the longest average hits and the second best accuracy.

	ASE	CASE	ET	Ours
Avg Hits	9.54 (5.94)	8.79 (5.28	6.55 (3.66)	**10.93 (6.28)**
Avg error	0.28 (0.33)	0.35 (0.39)	**0.25 (0.28)**	0.26 (0.31)

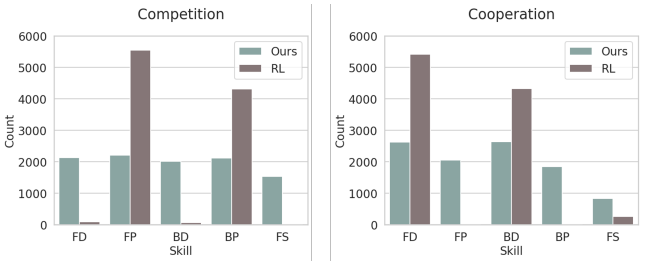

Figure 6: Skill command distribution of our method and RL.

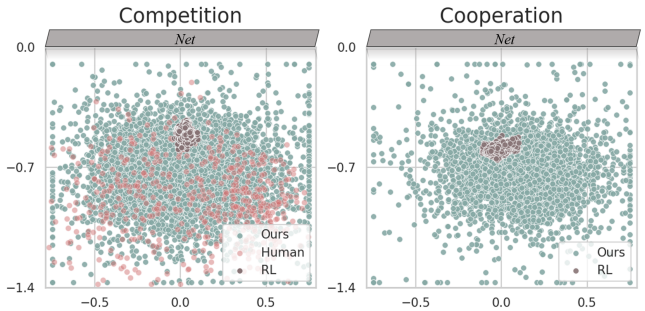

Figure 7: Target landing locations of our method, RL and Human.

7.2 Evaluation for agent-agent interaction

We evaluate the performance of learned strategies in the agent-agent interaction environment under both competition and cooperation settings. The competition strategy aims to develop an agent that achieves a higher winning rate than the opponent. The cooperation strategy develops an agent that can play gently with the opponent to increase the length of rallies. As a baseline, we learn a strategy policy via reinforcement learning (RL). Please refer to Appendix E for details on training the RL baseline. Our method and the RL baseline are compared by having them play with two types of opponents: the random strategy and the video strategy opponent introduced in Section 6. Each evaluation is computed over

Table 3: Strategy evaluation. We report the winning rates for the competition setting and average rounds for the cooperation setting.

	Competition		Cooperation	
	RL	Ours	RL	Ours
Random op	0.641	**0.687**	14.9	**16.4**
Video op	0.637	**0.681**	15.6	**18.2**

10k points. Table 3 shows the winning rate and the average rounds for the competition and cooperation settings. Our strategy learning algorithm can achieve higher winning rates for the competition setting and can maintain longer rallies for the cooperation setting for both opponents.

In Figure 6 and 7, we visualize the histogram of skill commands from the strategy policies, and the target landing locations. In Figure 7, we also provide ball landing locations captured from real players during competitive matches. We observe that our method has a more similar distribution of landing locations to humans than RL. RL converges to less diverse skill commands and it only hits to a small region of the table. In contrast, our method utilizes various skills and target locations throughout the gameplay. We also include qualitative gameplay visualizations in Figure 9. We further let RL and our method compete with each other, and report the winning rates in Table 4. Each method has two strategy policies trained with two opponents, therefore, we have four matches in total. Because RL falls into a local minimum and overfits to a specific opponent, our method achieves a much higher winning rate.

7.3 Evaluation of Human-agent interaction

Before learning strategies for the human-agent interaction environment, we finetune the skill-level controller using the play data of a human user interacting with the agent equipped with the original skill-level controller. The finetuning is required because of the domain gap between what the simulated agent has experienced and the styles of a real human user in the VR environment. After finetuning the skill controller, strategies are learned by following similar procedures as the agent-agent interaction environment. For training a competition strategy, we use demonstrations that result in the agent winning points, thus presenting more challenging returns for the human opponent. In contrast, for training a cooperative strategy, we use demonstrations where the human can maintain rallies, emphasizing easier ball returns for the human. These demonstrations serve as expert demonstrations in Algorithm 1. We report the winning rate of the agent and the average hits between the user and the agent in Table 5. When playing with the initial policies, the agent can achieve a winning rate of 64% and a rally with 4.04 hits on average. After two iterations of refinement of the competition strategy, the agent can achieve a winning rate of 78% , and the average number of hits drops to 3.75. For the cooperative strategy, the winning rate drops to 58%, and the user can achieve a rally with an average of 5.34 hits. These results demonstrate that our strategy learning algorithm is also effective for the human-agent interaction environment. We provide screenshots of real-time human-agent gameplay video in Figure 10.

Table 4: Winning rates between our method and RL. The opponent in parentheses is the opponent during training of the strategy policy.

	Ours (random op)	Ours (video op)
RL (random op)	0.45 vs 0.55	0.47 vs 0.53
RL (video op)	0.42 vs 0.58	0.42 vs 0.58

Table 5: Evaluation of human-agent interaction.

	Initial policies	Competition	Cooperation
Winning rate	0.64	0.78	0.58
Avg hits	4.04	3.75	5.34

8 DISCUSSION AND CONCLUSION

Although our method produces agents that play competitively and more naturally, it still has several limitations. First, although building individual policies for each skill and combining them via the mixer policy clearly improves the generated motion quality and task performance, our model would not scale well to a dataset including hundreds of different skills. Developing a hybrid model that combines our approach with a model learnable from unlabeled motions to achieve both high motion quality and scalability would be an interesting future research topic. Second, because our method is data-driven, the captured motion quality significantly affects the final motion quality. For example, the player tends to use large arm motions, and this motion style appears in our results as well. However, in matches, using less arm motion could be a way to conserve energy, and concealed movements can also confuse the opponent. Lastly, although we employ a rigid-body simulation for every component, including the ball, player, and table, where the ball can spin as well, air resistance is modeled using only damping based on velocity, rather than incorporating the Magnus effect, which bends the ball trajectory due to air pressure differences. This omission could impact the realism of our animations and the final strategies our system learned.

In this paper, we introduce a learning approach for physics-based table tennis animation. We develop a hierarchical controller structure, which overcomes the mode collapse problem that appears frequently in reusable latent-based models. Our approach not only improves overall motion quality but also enables us to learn effective decision strategies for two types of environments: agent-agent and human-agent interactions.

ACKNOWLEDGMENTS

This work was partially completed during Jiashun Wang's internship at The AI Institute. Jungdam Won was partially supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [NO.2021-0-01343-004, Artificial Intelligence Graduate School Program (Seoul National University)] and ICT(Institute of Computer Technology) at Seoul National University. We would like to thank Murphy Wonsick for helping to build the VR system and Melanie Danver for rendering the results.

REFERENCES

Jinseok Bae, Jungdam Won, Donggeun Lim, Cheol-Hui Min, and Young Min Kim. 2023. PMP: Learning to Physically Interact with Environments using Part-wise Motion Priors. In *ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023*. ACM, 64:1–64:10. https://doi.org/10.1145/3588432.3591487

Akhil Bagaria and George Konidaris. 2020. Option Discovery using Deep Skill Chaining. In *8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net. https://openreview.net/forum?id=B1gqipNYwH

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 (2019), 206:1–206:11. https://doi.org/10.1145/3355089.3356536

Dylan J. A. Brenneis, Adam S. R. Parker, Michael Bradley Johanson, Andrew Butcher, Elnaz Davoodi, Leslie Acker, Matthew M. Botvinick, Joseph Modayil, Adam White, and Patrick M. Pilarski. 2021. Assessing Human Interaction in Virtual Reality With Continually Learning Prediction Agents Based on Reinforcement Learning Algorithms: A Pilot Study. *arXiv preprint arXiv:2112.07774* (2021). https://arxiv.org/abs/2112.07774

Xuxin Cheng, Ashish Kumar, and Deepak Pathak. 2023. Legs as Manipulator: Pushing Quadrupedal Agility Beyond Locomotion. In *IEEE International Conference on Robotics and Automation, ICRA 2023*. IEEE, 5106–5112. https://doi.org/10.1109/ICRA48891.2023.10161470

Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. 2010. Feature-based locomotion controllers. *ACM Trans. Graph.* 29, 4 (2010), 131:1–131:10. https://doi.org/10.1145/1778765.1781157

Zhiyang Dou, Xuelin Chen, Qingnan Fan, Taku Komura, and Wenping Wang. 2023. C·ASE: Learning Conditional Adversarial Skill Embeddings for Physics-based Characters. In *SIGGRAPH Asia 2023 Conference Papers, SA 2023*. ACM, 2:1–2:11. https://doi.org/10.1145/3610548.3618205

Kevin French, Shiyu Wu, Tianyang Pan, Zheming Zhou, and Odest Chadwicke Jenkins. 2019. Learning Behavior Trees From Demonstration. In *International Conference on Robotics and Automation, ICRA 2019*. IEEE, 7791–7797. https://doi.org/10.1109/ICRA.2019.8794104

Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. 1995. Animating Human Athletics. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1995*. ACM, 71–78. https://doi.org/10.1145/218380.218414

Arushi Jain, Khimya Khetarpal, and Doina Precup. 2021. Safe option-critic: learning safety in the option-critic architecture. *Knowl. Eng. Rev.* 36 (2021), e4. https://doi.org/10.1017/S0269888921000035

Martin Klissarov, Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. Learnings Options End-to-End for Continuous Action Tasks. *arXiv preprint arXiv:1712.00004* (2017). http://arxiv.org/abs/1712.00004

George Dimitri Konidaris and Andrew G. Barto. 2009. Skill Discovery in Continuous Reinforcement Learning Domains using Skill Chaining. In *Advances in Neural Information Processing Systems 22*. 1015–1023. https://proceedings.neurips.cc/paper/2009/hash/e0cf1f47118daebc5b16269099ad7347-Abstract.html

Taesoo Kwon, Young-Sang Cho, Sang Il Park, and Sung Yong Shin. 2008. Two-Character Motion Analysis and Synthesis. *IEEE Trans. Vis. Comput. Graph.* 14, 3 (2008), 707–720. https://doi.org/10.1109/TVCG.2008.22

Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. 1996. Limit Cycle Control and Its Application to the Animation of Balancing and Walking. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996*. ACM, 155–162. https://doi.org/10.1145/237170.237231

Youngwoon Lee, Joseph J. Lim, Anima Anandkumar, and Yuke Zhu. 2021. Adversarial Skill Chaining for Long-Horizon Robot Manipulation via Terminal State Regularization. In *Conference on Robot Learning, 2021 (Proceedings of Machine Learning Research, Vol. 164)*. PMLR, 406–416. https://proceedings.mlr.press/v164/lee22a.html

Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S. Hu, and Joseph J. Lim. 2019. Composing Complex Skills by Learning Transition Policies. In *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net. https://openreview.net/forum?id=rygrBhC5tQ

Chengxi Li, Pai Zheng, Shufei Li, Yat Ming Pang, and Carman K. M. Lee. 2022. AR-assisted digital twin-enabled robot collaborative manufacturing system with human-in-the-loop. *Robotics Comput. Integr. Manuf.* 76 (2022), 102321. https://doi.org/10.1016/J.RCIM.2022.102321

C. Karen Liu, Aaron Hertzmann, and Zoran Popovic. 2006. Composition of complex optimal multi-character motions. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2006*. Eurographics Association, 215–222. https://doi.org/10.2312/SCA/SCA06/215-222

Huimin Liu, Zhiquan Wang, Christos Mousas, and Dominic Kao. 2020. Virtual Reality Racket Sports: Virtual Drills for Exercise and Training. In *2020 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2020*. IEEE, 566–576. https://doi.org/10.1109/ISMAR50242.2020.00084

Libin Liu and Jessica K. Hodgins. 2017. Learning to Schedule Control Fragments for Physics-Based Characters Using Deep Q-Learning. *ACM Trans. Graph.* 36, 3 (2017), 29:1–29:14. https://doi.org/10.1145/3083723

Libin Liu, Michiel van de Panne, and KangKang Yin. 2016. Guided Learning of Control Graphs for Physics-Based Characters. *ACM Trans. Graph.* 35, 3 (2016), 29:1–29:14. https://doi.org/10.1145/2893476

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. *ACM Trans. Graph.* 29, 4 (2010), 128:1–128:10. https://doi.org/10.1145/1778765.1778865

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. https://openreview.net/forum?id=fgFBtYgJQX_

Alejandro Marzinotto, Michele Colledanchise, Christian Smith, and Petter Ögren. 2014. Towards a unified behavior trees framework for robot control. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014*. IEEE, 5420–5427. https://doi.org/10.1109/ICRA.2014.6907656

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 *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net. https://openreview.net/forum?id=BJl6TjRcY7

Igor Mordatch, Emanuel Todorov, and Zoran Popovic. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.* 31, 4 (2012), 43:1–43:8. https://doi.org/10.1145/2185520.2185539

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. 2018. Self-Imitation Learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018 (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 3875–3884. http://proceedings.mlr.press/v80/oh18b.html

Stefan Pastel, Katharina Petri, C. H. Chen, Ana Milena Wiegand Cáceres, M. Stirnatis, C. Nübel, Lasse Schlotter, and Kerstin Witte. 2023. Training in virtual reality enables learning of a complex sports movement. *Virtual Real.* 27, 2 (2023), 523–540. https://doi.org/10.1007/S10055-022-00679-7

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 (2018), 143. https://doi.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 (2017), 41:1–41:13. https://doi.org/10.1145/3072959.3073602

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 Trans. Graph.* 41, 4 (2022), 94:1–94:17. https://doi.org/10.1145/3528223.3530110

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. AMP: adversarial motion priors for stylized physics-based character control. *ACM Trans. Graph.* 40, 4 (2021), 144:1–144:20. https://doi.org/10.1145/3450626.3459670

Mingyo Seo, Steve Han, Kyutae Sim, SeungHyeon Bang, Carlos Gonzalez, Luis Sentis, and Yuke Zhu. 2023. Deep Imitation Learning for Humanoid Loco-manipulation Through Human Teleoperation. In *22nd IEEE-RAS International Conference on Humanoid Robots, Humanoids 2023*. IEEE, 1–8. https://doi.org/10.1109/HUMANOIDS57100.2023.10375203

Hubert P. H. Shum, Taku Komura, Masashi Shiraishi, and Shuntaro Yamazaki. 2008. Interaction patches for multi-character animation. *ACM Trans. Graph.* 27, 5 (2008), 114. https://doi.org/10.1145/1409060.1409067

Hubert Pak Ho Shum, Taku Komura, and Shuntaro Yamazaki. 2012. Simulating Multiple Character Interactions with Collaborative and Adversarial Goals. *IEEE Trans. Vis. Comput. Graph.* 18, 5 (2012), 741–752. https://doi.org/10.1109/TVCG.2010.257

Richard S. Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.* 112, 1-2 (1999), 181–211. https://doi.org/10.1016/S0004-3702(99)00052-1

Eleven Table Tennis. 2016. https://elevenvr.com

Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. 2023. CALM: Conditional Adversarial Latent Models for Directable Virtual Characters. In *ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023*. ACM, 37:1–37:9. https://doi.org/10.1145/3588432.3591541

Kevin Wampler, Erik Andersen, Evan Herbst, Yongjoon Lee, and Zoran Popovic. 2010. Character animation in two-player adversarial games. *ACM Trans. Graph.* 29, 3 (2010), 26:1–26:13. https://doi.org/10.1145/1805964.1805970

Chao Wang, Anna Belardinelli, Stephan Hasler, Theodoros Stouraitis, Daniel Tanneberg, and Michael Gienger. 2023. Explainable Human-Robot Training and Cooperation with Augmented Reality. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems, CHI EA 2023*. ACM, 449:1–449:5. https://doi.org/10.1145/3544549.3583889

Jungdam Won, Deepak Gopinath, and Jessica K. Hodgins. 2020. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Trans. Graph.* 39, 4 (2020), 33. https://doi.org/10.1145/3386569.3392381

Jungdam Won, Deepak Gopinath, and Jessica K. Hodgins. 2021. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Trans. Graph.* 40, 4 (2021), 146:1–146:11. https://doi.org/10.1145/3450626.3459761

Jungdam Won, Deepak Gopinath, and Jessica K. Hodgins. 2022. Physics-based character controllers using conditional VAEs. *ACM Trans. Graph.* 41, 4 (2022), 96:1–96:12.

https://doi.org/10.1145/3528223.3530067

Pei Xu, Xiumin Shang, Victor B. Zordan, and Ioannis Karamouzas. 2023. Composite Motion Learning with Task Control. *ACM Trans. Graph.* 42, 4 (2023), 93:1–93:16. https://doi.org/10.1145/3592447

Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. 2022. ControlVAE: Model-Based Learning of Generative Controllers for Physics-Based Characters. *ACM Trans. Graph.* 41, 6 (2022), 183:1–183:16. https://doi.org/10.1145/3550454.3555434

KangKang Yin, Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2008. Continuation methods for adapting simulated skills. *ACM Trans. Graph.* 27, 3 (2008), 81. https://doi.org/10.1145/1360612.1360680

Haotian Zhang, Ye Yuan, Viktor Makoviychuk, Yunrong Guo, Sanja Fidler, Xue Bin Peng, and Kayvon Fatahalian. 2023. Learning Physically Simulated Tennis Skills from Broadcast Videos. *ACM Trans. Graph.* 42, 4 (2023), 95:1–95:14. https://doi.org/10.1145/3592408

Qingxu Zhu, He Zhang, Mengting Lan, and Lei Han. 2023. Neural Categorical Priors for Physics-Based Character Control. *ACM Trans. Graph.* 42, 6 (2023), 178:1–178:16. https://doi.org/10.1145/3618397

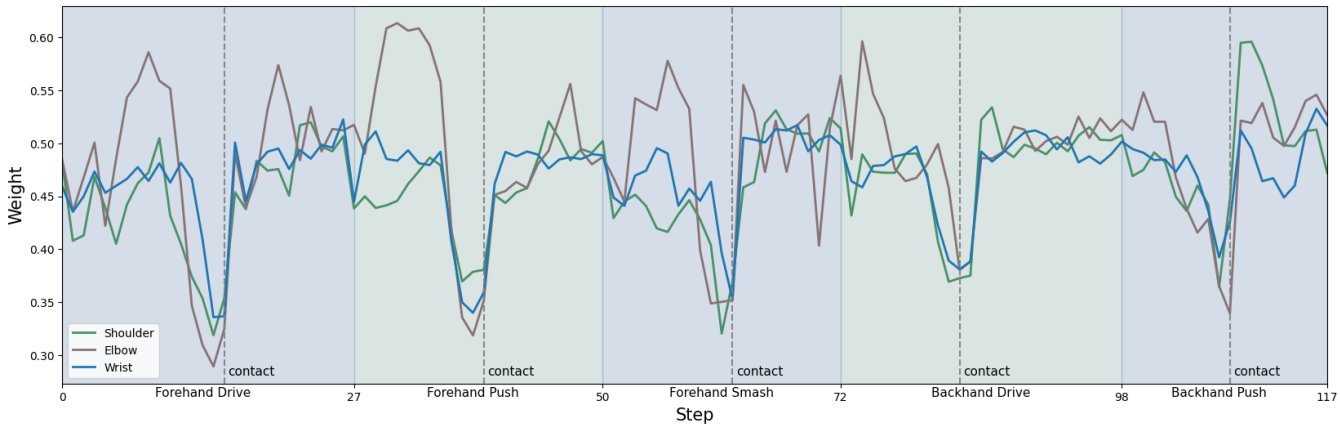

Figure 8: Visualization of the average blending weights φ of the shoulder, elbow, and wrist joints. The weights of the mixer policy are usually lowest when the paddle contacts the ball, and higher before and after transitions between different skills.

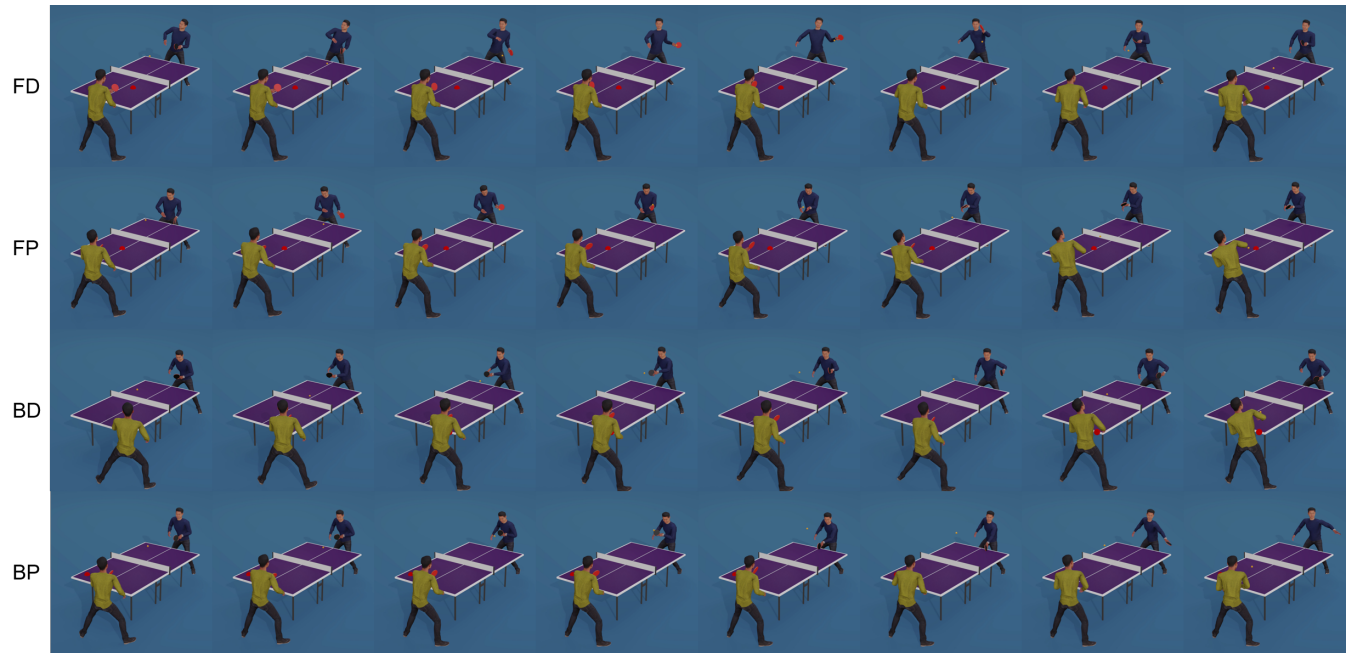

Figure 9: Agent-agent gameplay. Blue agent is applying our strategy-level controller. The red dot is the target. We demonstrate four skills; the forehand smash is less obvious because the opponent does not deliver high and slow shots.

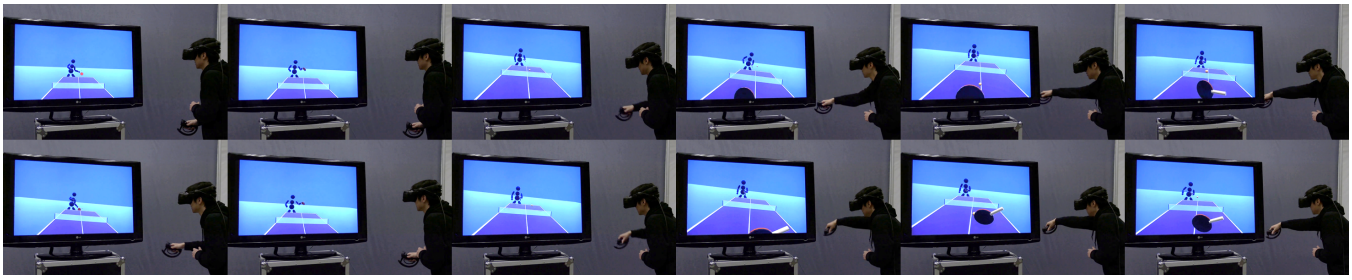

Figure 10: Human-agent interaction screenshots. A human controls a simulated paddle and the agent is simulated and controlled by our method.