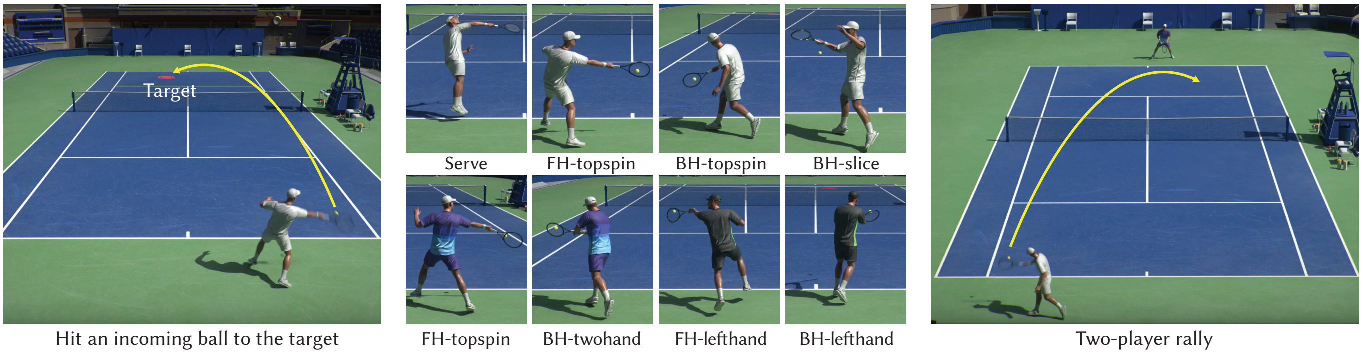
Learning Physically Simulated Tennis Skills from Broadcast Videos

HAOTIAN ZHANG, Stanford University, USA
YE YUAN, NVIDIA, USA
VIKTOR MAKOVIYCHUK, NVIDIA, USA
YUNRONG GUO, NVIDIA, Canada
SANJA FIDLER, NVIDIA, Canada, University of Toronto, Canada, and Vector Institute, Canada
XUE BIN PENG, Simon Fraser University, Canada and NVIDIA, Canada
KAYVON FATAHALIAN, Stanford University, USA



Hit an incoming ball to the target

Serve

FH-topspin

BH-topspin

BH-slice

FH-topspin

BH-twohand

FH-lefthand

BH-lefthand

Two-player rally

Fig. 1. We present a system that allows physically simulated characters to learn tennis skills from motions harvested from a large collection of broadcast tennis videos. Our simulated characters can hit the incoming ball to target positions accurately using a diverse array of strokes (serves, forehands, and backhands), spins (topspins and slices), and playing styles (one/two-handed backhands, left/right-handed play). Overall, our system can synthesize two physically simulated characters playing extended tennis rallies with simulated racket and ball dynamics.

We present a system that learns diverse, physically simulated tennis skills from large-scale demonstrations of tennis play harvested from broadcast videos. Our approach is built upon hierarchical models, combining a low-level imitation policy and a high-level motion planning policy to steer the character in a motion embedding learned from broadcast videos. When deployed at scale on large video collections that encompass a vast set of examples of real-world tennis play, our approach can learn complex tennis shotmaking skills and realistically chain together multiple shots into extended rallies, using only simple rewards and without explicit annotations of stroke types. To address the low quality of motions extracted from broadcast videos, we correct estimated motion with physics-based imitation, and use a hybrid control policy that overrides erroneous aspects of the learned motion embedding with corrections predicted by the high-level policy. We demonstrate that our system produces controllers for physically-simulated tennis players that can hit the incoming ball to target positions accurately using a diverse array of strokes (serves, forehands, and backhands), spins (topspins and slices), and playing styles (one/two-handed backhands, left/right-handed play). Overall, our system can synthesize two physically simulated characters playing extended tennis rallies with simulated racket and ball dynamics. Code and data for this work is available at https://research.nvidia.com/labs/toronto-ai/vid2player3d/.

CCS Concepts: • **Computing methodologies** → **Animation**.

Additional Key Words and Phrases: physics-based character animation, imitation learning, reinforcement learning

ACM Reference Format:
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 (August 2023), 14 pages. https://doi.org/10.1145/3592408

1 INTRODUCTION

Developing controllers for physics-based character simulation and control is one of the core challenges of computer animation. In recent years, techniques that combine deep reinforcement learning (DRL) and motion imitation have produced simulated characters that exhibit impressive lifelike motions and perform a range of athletic skills. The vast majority of systems use motion capture (mocap) data as the source of kinematic motions to imitate. Unfortunately, it is costly to acquire large amounts of high quality mocap animation. In contrast, video of athletic events is widely available and provides a

Authors' addresses: Haotian Zhang, haotianz@stanford.edu, Stanford University, USA, Stanford; Ye Yuan, yey@nvidia.com, NVIDIA, USA, Santa Clara; Viktor Makoviychuk, vmakoviychuk@nvidia.com, NVIDIA, USA, Santa Clara; Yunrong Guo, kellyg@nvidia.com, NVIDIA, Canada, Toronto; Sanja Fidler, sfidler@nvidia.com, NVIDIA, Canada, Toronto and University of Toronto, Canada, Toronto and Vector Institute, Canada, Toronto; Xue Bin Peng, xbpeng@sfu.ca, Simon Fraser University, Canada, Vancouver and NVIDIA, Canada, Vancouver; Kayvon Fatahalian, kayvonf@cs.stanford.edu, Stanford University, USA, Stanford.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0730-0301/2023/8-ART $15.00
https://doi.org/10.1145/3592408

rich source of in-activity motion data. Long running video streams provide examples of the full range of skills an athlete must perform in a sport: not just salient actions (hitting a shot in tennis), but the complex and subtle motions athletes use to transition between these movements. Furthermore, the ability to acquire large amounts of video allows examples of many variations of each action to be observed (hitting a high ball or a low ball, reacting quickly or slowly).

In this paper, we ask the following questions: (1) how we can leverage large-scale, but lower-quality, databases of 3D tennis motion harvested from broadcast videos of professional play to produce controllers that can accomplish a challenging athletic task – playing full tennis rallies with simulated racket and ball dynamics; (2) How we can leverage state-of-the-art methods in data-driven and physically-based character animation to help learning skills from video data; (3) How we can learn character controllers with a diverse set of skills without explicit skill annotations, such as hitting different types of shots (serves, forehands, and backhands), employing different spins (topspins and slices), and recovering to prepare for the next shot.

Our approach builds upon recent ideas in hierarchical physics-based character control: leveraging motions produced by physics-based imitation of example videos to learn a rich motion embedding for tennis actions, and then training a high-level motion controller that steers the character in the latent motion space to achieve higher-level task objectives (e.g., hitting an incoming tennis ball), with low-level movements controlled by the imitation controller. To address motion quality issues caused by perception errors that persist in the learned motion embedding (e.g., blurred or occluded wrist motion, inaccurate neck rotations), our pipeline overrides erroneous reference motion with physics-based corrections driven by high-level task rewards or by using simple kinematic constraints specific to tennis (e.g. players should keep their eye on the ball). Our system utilizes residual force control [Yuan and Kitani 2020] to increase overall task performance, but can also function without residual forces while incurring only modest task performance reductions.

We demonstrate controllers for physically-simulated tennis players that can hit the ball to target positions on the court with high accuracy and can successfully conduct competitive rally play that includes a range of shot types and spins, as shown in Figure 1. Specifically, the contributions of this work are as follows:

- **Learning diverse and complex tennis skills from broadcast videos.** We present a video imitation system built upon hierarchical models. The system combines a low-level imitation policy and a high-level motion planning policy to steer the character in a motion embedding learned from large video datasets, so that complex skills such as hitting tennis balls with different types of shots and spins can be learned with simple rewards and without explicit annotations of these action types.
- **Motion reconstruction pipeline for generating motion embeddings with higher motion quality using physics priors.** We develop a full pipeline to reconstruct physically plausible tennis motion from monocular broadcast videos using physics-based imitation. Constructing the motion embedding based on physically corrected motions leads to more natural motions and better end task performance than training an embedding directly from the results of kinematic pose estimators without physics correction.
- **Hybrid approach for building motion controllers from imperfect motion data.** We propose a hybrid approach for complementing motion reconstruction from videos with RL-based skill learning. We overcome artifacts in hard-to-perceive motions (e.g. blurred/occluded wrist motion) using corrections predicted by a high-level policy aiming to accomplish a high-level tennis task.

2 RELATED WORK

We review previous research from the domains of monocular human pose/motion estimation to physics-based character control. We also discuss prior work on learning skills from videos and summarize prior work in synthesizing character animation for sports.

Monocular human pose/motion estimation. Estimating 3D human motion from monocular video has been a long-standing problem in computer vision. Kinematic techniques have seen remarkable advances with the use of Convolutional Neural Networks (CNNs) to predict 3D joint locations from images using supervised learning [Dabral et al. 2018; Hossain and Little 2018; Mehta et al. 2018, 2017; Pavllo et al. 2019]. Statistical human body models, such as SMPL [Loper et al. 2015], provide priors for simultaneously solving joint positions and recovering body meshes. It can be done either through optimization to fit 2D keypoints detection [Bogo et al. 2016; Lassner et al. 2017] or via direct regression using neural networks [Guler and Kokkinos 2019; Kanazawa et al. 2018; Omran et al. 2018; Pavlakos et al. 2018; Tung et al. 2017]. While most of the algorithms work on single images, some systems have taken advantage of the neighboring frames to improve temporal stability [Arnab et al. 2019; Huang et al. 2017; Kanazawa et al. 2019; Kocabas et al. 2020; Sun et al. 2019]. Recently, HybrIK [Li et al. 2021] employs inverse kinematics to estimate the human meshes with absolute translations in the camera coordinates. GLAMR [Yuan et al. 2022] addresses the global pose estimation problem via global root trajectory optimization. Although kinematic pose estimation is able to provide visually plausible results, they often produce physically impossible motions with artifacts such as jitter, foot sliding, and ground penetration.

A number of prior works have attempted to mitigate nonphysical artifacts arising from kinematic pose estimators by incorporating physical constraints and human dynamics. Many of the prior techniques [Brubaker et al. 2009; Rempe et al. 2020; Shimada et al. 2020; Vondrak et al. 2012a; Wei and Chai 2010; Zell et al. 2017] use trajectory optimization to find physical forces that reproduce the given motion depicted in a video clip. Neural PhysCap [Shimada et al. 2021] embeds hard physics constraints through a custom neural network layer to enable fully differentiable supervised learning. In this work, our goal is not only to estimate motions from video clips, but also to leverage the motion data to synthesize controllers that enable physically simulated characters to perform highly dynamic sports activities.

Physics-based character control. Building controllers for physics-based character simulation and control is one of the core challenges of computer animation. Early approaches have leveraged optimization techniques with hand-crafted control structures, such as finite state machines, to create controllers for a large variety of complex behaviors [Coros et al. 2010; De Lasa et al. 2010; Hodgins et al. 1995; Levine and Koltun 2013; Liu et al. 2012; Mordatch et al. 2012; Raibert and Hodgins 1991; Tan et al. 2014; Yin et al. 2008, 2007]. More recently, deep reinforcement learning (DRL) has been used to drastically expand the corpus of skills that can be reproduced by simulated characters, including locomotion [Peng et al. 2017; Schulman et al. 2015; Xie et al. 2020; Yu et al. 2018], dressing [Clegg et al. 2018], object manipulation [Merel et al. 2020], and recovery behaviors [Tao et al. 2022]. The difficulty of designing reward functions that lead to lifelike motions has motivated the use of motion imitation techniques, where naturalistic motion can be learned by imitating reference motion data, either through explicit motion tracking [Lee et al. 2010; Liu et al. 2016, 2010; Peng et al. 2018a] or adversarial imitation learning [Peng et al. 2022, 2021]. Low-level motion imitation controllers can be combined with high-level motion synthesis models to form a hierarchical framework used to direct a character to accomplish more complex tasks which may require composition of multiple skills in furtherance of a desired task objective. Bergamin et al. [2019] use motion-matching to generate plausible motion trajectories for guiding a physically simulated character to perform a given task, while Park et al. [2019] train an autoregressive motion prediction model to generate plausible human motions. Won et al. [2022] and Yao et al. [2022] train a VAE-like control model using model-based RL with a learned differentiable world model. Once trained, the VAE latent space can be used in a control hierarchy to direct a character to perform new downstream tasks. Our proposed video imitation system is also built upon the hierarchical models, where a motion embedding is constructed using a conditional VAE. The learned motion embedding can be used to synthesize reference motion that achieves the given task by searching a sequence of latent codes with DRL, similar to Motion VAE [Ling et al. 2020]. Instead of using mocap data, our system learns physics-based character controllers directly from broadcast videos.

Learning skills from videos. Mocap has been the most commonly used source of motion data for physics-based character animation. While mocap can provide high-quality motion data, these systems typically require heavy instrumentation of both the environment and actors, which can be difficult to apply to large-scale outdoor sports. Video clips can provide a much more abundant and accessible source of motion data. Vondrak et al. [2012b] represents one of the first physics-based character animation systems for video imitation, using hand-designed FSM controllers and an incremental optimization approach with a 2D-silhouette matching objective. Peng et al. [2018b] combine 2D/3D pose estimators and deep reinforcement learning to train controllers to imitate a diverse corpus of skills from short video clips. Yu et al. [2021] extend this approach to imitate longer video sequences with dynamic camera movements and irregular environments. Yuan and Kitani [2020] propose residual force control to compensate for the dynamics mismatch between simulated characters and real humans, which enables tracking of agile motions such as ballet dance. Most recently, SimPoE [Yuan et al. 2021] trained general motion tracking controllers using large-scale mocap dataset (e.g, AMASS [Mahmood et al. 2019]), which enables zero-shot imitation of skills from monocular videos in real-time. In this work, we adopt an approach similar to SimPoE for controlling the low-level behaviors of the simulated character and generating physically corrected motion from low-quality estimated kinematic motion. Unlike prior systems [Peng et al. 2018b; Yu et al. 2021] which train individual controllers for each short video clip, our system learns more diverse and versatile tennis skills by imitating large-scale video datasets, which enables characters to synthesize agile motions for more complex long-horizon tasks, such as extended tennis rallies between two players.

Character animation for sports. Sports have long been a popular testbed for character animation. The seminal work of Hodgins et al. [1995] design state machine controllers to reproduce a variety of Olympic sports, including running, bicycling, and vaulting. Zordan and Hodgins [2002] combine physics-based simulation with motion capture data to simulate responsive behaviors for boxing, fencing, and table tennis. Van de Panne and Lee [2003] use Proportional Derivitive (PD) controllers to develop an interactive simulation for ski stunts. More recent work leveraging reinforcement learning and trajectory optimization have demonstrated impressive results for training controllers for more complex athletic skills, such as basketball dribbling [Liu and Hodgins 2018], soccer dribbling and shooting [Hong et al. 2019], figure skating [Yu et al. 2019], multi-agent soccer [Liu et al. 2021], boxing and fencing [Won et al. 2021], and soccer juggling [Xie et al. 2022]. It is notable that the vast majority of these prior systems utilize motion capture data, which can be scarce for highly-specialized athletic activities. In this work, we aim to utilize widely available video data to develop versatile controllers for sports, which also enables models to capture different styles when using different players' video clips.

Finally, our work is inspired by Vid2Player [Zhang et al. 2021], which uses broadcast tennis videos to create a system to synthesize 2D player sprites that behave and appear like professional tennis players. Our work generates physically simulated 3D character animation which can be viewed and rendered from any viewpoint.

3 OVERVIEW

An overview of our system is shown in Figure 2. Our system takes as input unannotated broadcast tennis videos of different players, and outputs controllers for physically simulated characters that hit consecutive incoming tennis balls using a diverse set of tennis skills. The controllers can be used to produce 3D character animation depicting two simulated characters playing tennis rallies.

Our system consists of four stages. First, we estimate 2D and 3D player poses and global root trajectories to create a kinematic motion dataset (Section 4). Second, a low-level imitation policy is trained to imitate the kinematic motion for controlling the low-level behaviors of the simulated character and generate a physically corrected motion dataset (Section 5). Next, we fit a conditional VAE to the corrected motion dataset to learn a low-dimensional motion embedding that produces human-like tennis motions (Section 6). Finally, a high-level motion planning policy is trained to generate

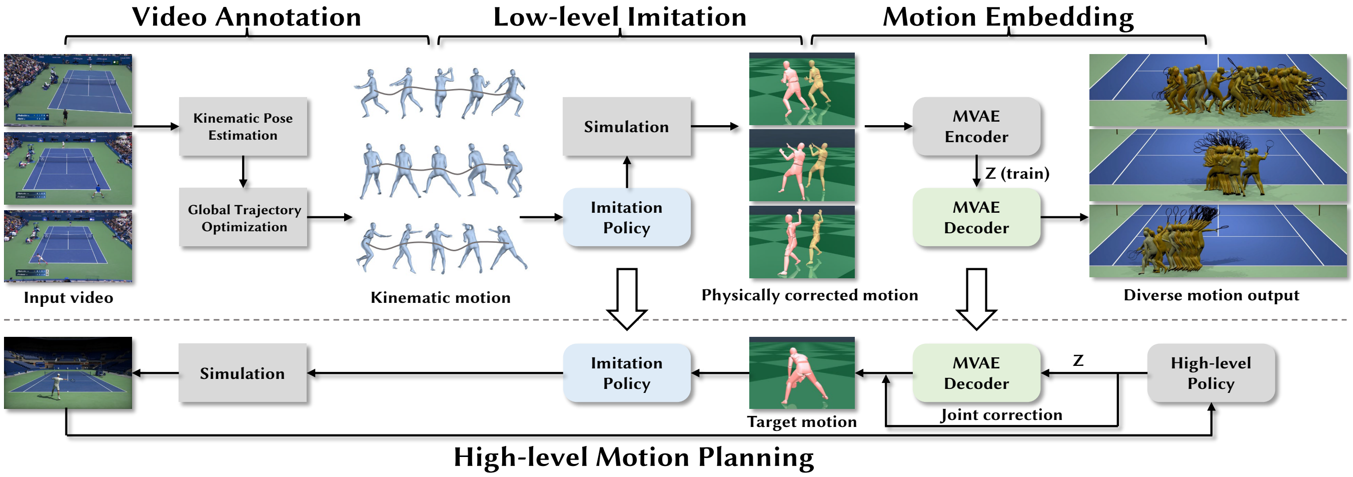


Fig. 2. Our video imitation system consists of four stages: First, we estimate kinematic motions from source video clips. Second, a low-level imitation policy is trained to imitate the kinematic motion for controlling the low-level behaviors of the simulated character and generate physically corrected motion. Next, we fit a conditional VAE to the corrected motion to learn a motion embedding that produces diverse and human-like tennis motions. Finally, a high-level motion planning policy is trained to generate target kinematic motion by predicting VAE latent codes and joint corrections for wrist motion. The target motion is then imitated by the low-level policy to control a physically simulated character to perform the desired task.

target kinematic motion by combining body motion output from the motion embedding with predicted corrections for the character's wrist motion. The target motion is then imitated by the low-level policy to control a physically simulated character to perform the desired task (Section 7).

4 VIDEO ANNOTATION

We collect 13 US Open match videos (1080p), provided courtesy of the Tennis Channel [TennisChannel 2023]. The videos consist of matches of three players Roger Federer, Rafael Nadal, and Novak Djokovic playing against each other or against other players, which took place between 2017-2021. To build our tennis motion dataset from the raw match videos, we utilize automatic machine annotations to estimate players' kinematic motions from the main broadcast camera view and manual annotations for players' identities and racket-ball contact times. In total, our motion dataset contains 80 minutes of motion data from Federer, 96 minutes from Djokovic, and 103 minutes from Nadal. The three players are chosen because they appeared most frequently in the selected time range. All video clips are recorded in 30 fps. Next, we describe the details of the machine annotations and manual annotations.

Player tracking and pose estimation. We track the players and estimate their 2D/3D poses from the broadcast videos by using off-the-shelf detection models. We run Yolo4 [Bochkovskiy et al. 2020] to track players on both sides of the court to obtain point boundaries and player bounding boxes. 2D pose keypoints are extracted using ViTPose [Xu et al. 2022]. Finally, we use HybrIK [Li et al. 2021] to estimate the body shape and pose parameters for SMPL [Loper et al. 2015]. Player bounding boxes are used to crop the images around the player before being provided as input to the pose estimator.

Global root trajectory and camera estimation. Since HybrIK only outputs root position and orientation in camera coordinates, those quantities need to be converted to the global court coordinates, where the origin is located at the center of the court. We first estimate the camera projection using the method from Farin et al. [2003] to detect court lines and their intersections, and then solve for the camera matrix with the Perspective-N-Point algorithm. The camera transformation can then be used to obtain the global root orientation. To estimate the player's global root position, instead of using the translation from the camera transformation, we first compute the 2D position of the player's root projected onto the ground (center of two ankle keypoints), then transform the location into court coordinates with the inverse camera projection. We further correct the root trajectory by solving an optimization problem similar to GLAMR [Yuan et al. 2022] to minimize the re-projection error between 2D keypoints and projected 3D joint positions. We only estimate the motion of the player on the near side of the court since this player is larger in the frame. Pose and depth estimates for the player on the far side of the court are less reliable. The kinematic motion dataset obtained from this stage is referred to as $\mathbb{M}_{kin}$.

Manual annotations. To facilitate modeling the phase of the tennis motion when learning the motion embedding (Section 6), we manually label the frames where a player makes ball contact. Although we chose to manually label ball contact times for this project, it is possible to automate this labeling process with high accuracy using modern computer vision techniques [Hong et al. 2022]. To learn the styles of different players, we also manually annotate the identity of the paper in each motion sequence.

5 LOW-LEVEL IMITATION POLICY

Since the estimated kinematic motion is obtained without explicit modeling of human dynamics, it will contain physically implausible motions such as jitter, foot skating, and ground penetration. To correct these artifacts, we train a low-level imitation policy to control a physically simulated character to track this noisy kinematic motion and output physically corrected motion. In addition to its use for motion reconstruction, the low-level policy is also used to control the low-level movements of the simulated character to perform new tasks by tracking the target kinematic trajectories from the high-level motion planning policy (Section 7).

The approach we use is similar to SimPoE [Yuan et al. 2021]. The task of controlling the character agent in a physically simulated environment to mimic reference motions can be formulated as a Markov decision process (MDP), defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$ of states, actions, transition dynamics, a reward function, and a discount factor. We first initialize the state of the simulated character $\mathbf{s}_0$ to be the same initial state of the reference motion. Starting from s_0, the agent iteratively samples actions $\mathbf{a}_t \in \mathcal{A}$ according to a policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$ at each state $\mathbf{s}_t \in \mathcal{S}$. The environment then transitions to the next state $\mathbf{s}_{t+1}$ according to the transition dynamics $\mathcal{T}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, and then outputs a scalar reward r_t for that transition. The reward is computed based on how well the simulated motion aligns with the reference motion. The goal of this learning process is to learn an optimal policy π^* that maximizes the expected return $J(\pi) = \mathbb{E}_\pi[\Sigma_t \gamma^t r_t]$. Next, we describe the details of the state, action, and reward function, as well as training strategy of the low-level policy.

States. The simulated character model is created based on the SMPL format [Loper et al. 2015], with body shape parameters estimated using HybrIK [Li et al. 2021]. The character consists of 24 rigid bodies and 72 degrees of freedom. We use the following features to represent the character state $\mathbf{s}_t = (\mathbf{p}_t, \dot{\mathbf{p}}_t, \mathbf{q}_t, \dot{\mathbf{q}}_t, \hat{\mathbf{p}}_{t+1}, \hat{\mathbf{q}}_{t+1})$:

- $\mathbf{p}_t$: joint positions in the character's root coordinates
- $\dot{\mathbf{p}}_t$: joint linear velocities in the character's root coordinates
- $\mathbf{q}_t$: joint rotations in the joints' local coordinates
- $\dot{\mathbf{q}}_t$: joint angular velocities in the joints' local coordinates
- $\hat{\mathbf{p}}_{t+1}$: target (kinematic) joint positions
- $\hat{\mathbf{q}}_{t+1}$: target (kinematic) joint rotations

Actions. Similar to many prior systems, we use proportional derivative (PD) controllers at each non-root joint to produce torques for actuating the character's body. The action $\mathbf{a}_t$ specifies the target joint angles $\mathbf{u}_t$ for the PD controllers. At each simulation step, the joint torques τ_t are computed as:

$$\tau_t = \mathbf{k}_p \cdot (\mathbf{u}_t - \mathbf{q}_t^{nr}) - \mathbf{k}_d \cdot \dot{\mathbf{q}}_t^{nr} \, , \tag{1}$$

where $\mathbf{k}_p$ and $\mathbf{k}_d$ denote the parameters of the PD controllers that determine the stiffness and damping of each joint, $\mathbf{q}_t^{nr}$ and $\dot{\mathbf{q}}_t^{nr}$ are the joint rotations and angular velocities of the non-root joints. To improve tracking performance on highly agile motions, we also allow the policy to apply external residual forces to the root [Yuan and Kitani 2020]. Therefore, the actions also include residual forces and torques η_t for the root joint, and each action is defined as $\mathbf{a}_t = (\mathbf{u}_t, \eta_t)$.

Rewards. The reward function is designed to encourage the policy to closely track the reference motion while also minimizing energy expenditure. The reward consists of five terms:

$$r_t = \omega_o r_t^o + \omega_v r_t^v + \omega_p r_t^p + \omega_k r_t^k + \omega_e r_t^e \, . \tag{2}$$

The joint rotation reward r_t^o measures the difference between the local joint rotations of the simulated character $\mathbf{q}_t^j$ and the reference motion $\hat{\mathbf{q}}_t^j$:

$$r_t^o = \exp\left[-\alpha_o \sum_j \left(||\mathbf{q}_t^j \ominus \hat{\mathbf{q}}_t^j||^2 \right) \right], \tag{3}$$

where $\ominus$ denotes the geodesic distance between two rotations.

The velocity reward r_t^v measures the mismatch between local joint velocities of the simulated motion $\dot{\mathbf{q}}_t^j$ and the reference motion $\hat{\dot{\mathbf{q}}}_t^j$:

$$r_t^v = \exp\left[-\alpha_v \sum_j \left(||\dot{\mathbf{q}}_t^j - \hat{\dot{\mathbf{q}}}_t^j||^2 \right) \right]. \tag{4}$$

The joint position reward r_t^p encourages the 3D world joint positions $\mathbf{x}_t^j$ (including the root joint) to match the reference motion $\hat{\mathbf{x}}_t^j$:

$$r_t^p = \exp\left[-\alpha_p \sum_j \left(||\mathbf{x}_t^j - \hat{\mathbf{x}}_t^j||^2 \right) \right]. \tag{5}$$

The keypoint reward r_t^k encourages the projected 2D joint positions $\bar{\mathbf{x}}_t^j$ to match the detected 2D keypoints $\tilde{\mathbf{x}}_t^j$:

$$r_t^k = \exp\left[-\alpha_k \sum_j \left(||\bar{\mathbf{x}}_t^j - \tilde{\mathbf{x}}_t^j||^2 \right) \right]. \tag{6}$$

Finally, the reward r_t^e denotes the power penalty computed as:

$$r_t^e = -\sum_j \left(||\dot{\mathbf{q}}_t^j \cdot \tau_t^j||^2 \right), \tag{7}$$

where τ_t^j is the internal torque applied on the joint j. The weight and scale factor for each reward term is manually specified and kept the same in all the experiments (see supplementary material).

Training. The training of the low-level policy is conducted in two stages. In the first stage, we train the policy with a high-quality mocap database AMASS [Mahmood et al. 2019] to learn to imitate general motions. However, directly applying this policy to track the tennis motion from $\mathbb{M}_{kin}$ will lead to the character falling after a few steps of simulation since the noisy tennis motions differ from the motion examples in AMASS. Therefore, we find it necessary to fine-tune the policy using $\mathbb{M}_{kin}$ so that it can track the tennis motion more closely without falling. The power penalty r_t^e is only applied during this fine-tuning stage to mitigate the impact of frame-to-frame variances in $\mathbb{M}_{kin}$. Once trained, we can simply run the low-level policy to track each motion sequence from $\mathbb{M}_{kin}$ and export the physically corrected tennis motion dataset, referred as $\mathbb{M}_{corr}$. More details of the rewards, network architecture, and training hyper-parameters can be found in the supplementary material.

As mentioned above, the training process yields solutions that rely on residual forces. Reducing this reliance is a challenging problem, but stands to yield more physically accurate motion. To reduce this reliance, we can optionally fine-tune the policy by gradually decreasing the allowed maximum scale of residual forces/torques η_t to zero during training. We evaluate the motion quality and task performance effects of removing residual forces in Section 9.8.

6 MOTION EMBEDDING

Given $\mathbb{M}_{corr}$, we build a kinematic motion embedding to serve as the action space for subsequent high-level planning of long-term tennis motions. This generative model is instantiated using conditional VAE, which learns a low-dimensional latent space using $\mathbb{M}_{corr}$.

Our motion embedding model is based on the motion VAE(MVAE) model [Ling et al. 2020]. Given the character's current pose and a latent variable z representing possible transitions from the current pose, MVAE reconstructs the pose in the next time step while shaping z into a normal distribution. At run-time, the encoder is discarded and the decoder takes the input of the current pose and a latent z to produce the next pose. The predicted next pose can be used as the input in the next step to generate a sequence of poses autoregressively. Our system adapts the original MVAE in two ways: (1) to prevent global drift, we condition on pose features in global court coordinates and (2) in addition to the pose, our model also predicts the phase of the character's motion (tennis motion is cyclic from shot to shot). Phase information simplifies the design of the reward functions of the high-level policy.

Pose representation. The pose in each frame is represented using the following features:

- $\mathbf{q}_t^r$: root orientation in the global court coordinates
- $\mathbf{r}_t$: root position in the global court coordinates
- $\dot{\mathbf{r}}_t$: root linear velocity in the global court coordinates
- $\mathbf{p}_t$: joint positions relative to the root in global court coordinates
- $\dot{\mathbf{p}}_t$: linear joint velocities in the global court coordinates
- $\mathbf{q}_t$: joint rotations in the local joint coordinates

Representing positions/orientations in the global court coordinates provide a strong prior to the tennis motion. For example, backhand motions are more likely to be performed on the left side of the court for a right-handed player and players should be facing toward the net after each shot. In practice, we find that conditioning motion generation on global positions/orientations yields motion that more consistently recovers back toward the center of the court and has the character face the net as they ready themselves for the next shot.

Motion phase. Tennis players undergo cyclic motion from shot-to-shot during a point. Knowing the current phase of this motion simplifies the design of reward functions for the high-level policy which is responsible for long-term motion planning. For example, a reward can be designed to minimize the distance between the racket and the ball at ball contact time. Therefore, we adapt MVAE to also predict the motion phase for the output pose. Specifically, we represent the motion phase at each frame with a cyclic phase variable θ in $[0, 2\pi]$ based on the shot-cycle state machine from Vid2Player [Zhang et al. 2021]. $\theta = \pi$ denotes when the player makes ball contact and $\theta = 0$ *or* $\theta = 2\pi$ denotes when the player recovers (the opponent makes ball contact). The phase for the rest of the frames is linearly interpolated between the neighboring two anchors. To avoid a discontinuity at $\theta = 2\pi$, we encode the motion phase with $\sin\theta$ and $\cos\theta$. Due to the repetitive structure of tennis motion, the motion phase can be reliably learned in a semi-supervised setting by providing phase information for a sparse sampling of shots (we annotated only 20% of all shots) We find that the model's predicted phase is always close to π when the corresponding swing motion nears the point where the ball should be contacted.

Training. We follow the network design and general training setup of MVAE [Ling et al. 2020] and incorporate a number of strategies that are crucial to successfully training a model on the reconstructed tennis motions. Since the input motion data from $\mathbb{M}_{corr}$ is still noisier than mocap data, MVAE tends to be more susceptible to error accumulation when generating longer sequences at run-time. To improve the stability of the autoregressive predictions, we follow Ling et al. [2020] and adopt *scheduled sampling* [Bengio et al. 2015]. In our experiments, we find that the selection of the coefficient β for the KL divergence loss is critical for learning a good motion embedding for use by the high-level policy. When β is too large, the decoder will ignore the latent variable z and only playback the original motion data. When β is too small, MVAE may overgeneralize and produce implausible motions with clear artifacts such as foot skate. Empirically, we find that $\beta = 0.5$ effectively balances the flexibility and motion quality of the learned motion embedding. More details of the training process are provided in the supplementary material.

7 HIGH-LEVEL MOTION PLANNING POLICY

Given a motion embedding capable of producing a diverse set of tennis motions, we train a high-level motion planning policy that synthesizes novel motions that enable a character to perform tasks such as hitting an incoming tennis ball to specific target locations. The high-level policy selects latents from the motion embedding to generate kinematic motion trajectories that resemble human behaviors [Ling et al. 2020]. The resulting kinematic motions are then used as target reference trajectories to drive a physically simulated character using the low-level imitation policy trained in Section 5.

However, directly applying the aforementioned approach will fail to produce characters that successfully hit the ball back into the court with a high degree of success. The problem is that even with physics correction, occlusions and motion blur result in $\mathbb{M}_{corr}$ containing notable errors in the character's estimated wrist motion (and corresponding racket motion). Even small errors in swing motion can prevent the high-level policy from finding motion solutions that accurately hit the ball. To overcome inaccuracies in the reconstructed motion data, we propose a hybrid control approach where the full-body motion is controlled by the reference trajectories generated by the MVAE, while the wrist motion is directly controlled by the high-level policy. We optimize the high-level policy using a curriculum curated for tennis play.

In addition to swing motion errors, the reconstructed tennis motions feature additional errors such as the character's eyes not tracking the ball and the character's non-dominant hand not gripping the racket during two-handed swings. Since addressing these artifacts would require additional reward engineering, we propose alternative solutions using simple kinematic constraints specific to tennis.

7.1 Policy Representation

The problem of jointly optimizing the predicted MVAE latent codes and predicted joint corrections can be formulated as an MDP and solved with reinforcement learning. We now describe the details of the state and action representations used for the high-level policy.

States. The state consists of a set of features that describes the state of the character and the incoming ball, as well as control targets specified by the system. The character state shares the same pose representation used for the MVAE from Section 6, however all features are now computed from the simulated character instead of the kinematic motion. The ball state is represented using the ball's position in the next ten frames (including the current position), which provides the policy with a forecast of the ball's future trajectory. The future trajectory of the ball is estimated given the ball's launch velocity, spin, and height (more details can be found in the supplementary material). Control targets consist of the desired placement of the character's next shot (the position where the ball should bounce on the other side of the court) and a binary variable indicating the desired spin direction of the next shot (topspin or backspin).

Actions. Each action consists of two components: a latent code for MVAE to generate a kinematic target pose for the next frame, and joint corrections for the swing arm. The joint corrections include three Euler angles: two for the wrist joint (excluding the twist angle since the twist is limited for the wrist), and the twist angle for the elbow joint. The joint corrections overwrite the rotations from the MVAE-produced pose, and the final corrected pose is used as the target pose for the low-level imitation policy to track.

7.2 Reward Function

We apply our framework to train control policies that enable the simulated character to hit an incoming tennis ball so that it bounces at desired location on the court (ball bounce position) and with a target spin direction. This objective is represented using two reward functions specified for stages before and after the racket-ball contact. Before contact, we apply the racket position reward r_t^r to minimize the distance between the center of the racket head $\mathbf{x}_t^r$ and the ball position $\mathbf{x}_t^b$ when the character hits the ball (predicted motion phase θ_t gets close to π).

$$r_t^r = \exp(-\alpha_r||\mathbf{x}_t^r - \mathbf{x}_t^b||^2) \cdot \exp(-\alpha_\theta||\theta_t - \pi||^2), \tag{8}$$

where α_r and α_θ are scaling factors. After contact, we apply the ball position reward r_t^b to minimize the distance between the estimated ball bounce position $\tilde{\mathbf{x}}^b$ and the target bounce position $\hat{\mathbf{x}}^b$ while ensuring the ball spins in the right direction as the target spin direction.

$$r_t^b = \begin{cases} 0 & \text{if } s^b \neq \hat{s}^b \\ \exp(-\alpha_b||\tilde{\mathbf{x}}^b - \hat{\mathbf{x}}^b||^2) & \text{if } s^b = \hat{s}^b \end{cases}, \tag{9}$$

where s^b and $\hat{s}^b$ are binary variables that represent the simulated and target ball spin direction, respectively. A value of 1 denotes topspin (the ball spins forward) and a value of 0 denotes backspin (the ball spins backward). Since different strokes are needed to generate different spins to match $\hat{s}^b$, this reward causes the high-level policy to produce the appropriate stroke type even though the strokes in the source video are not annotated. At the moment of racket-ball contact, we immediately estimate $\tilde{\mathbf{x}}^b$ and apply the same r_t^b at every time step after contact. The two rewards are used differently at different curriculum stages described in the next section.

7.3 Training

We design the training strategy as follows. At the beginning of each episode, the character is initialized at a random court position near the baseline in a ready pose. The incoming balls are launched every 2-2.5 seconds from positions near the baseline of the opponent's side of the court, with a launch velocity between 25-35 m/s, and a launch spin between 0-50 RPS. The ball can bounce anywhere between the service line and the baseline of the character's side of the court, which covers a wide variety of incoming ball trajectories. To train the character to serve, we also initialize the character to a pre-service state and initialize the ball to be thrown into the air at the beginning of the training episode. The maximum episode length is set to be 300 frames (10 seconds) which allows the character to practice four consecutive shots in each episode. We found that simulating multiple shots per episode leads to better performance compared to only one shot per episode.

Curriculum Learning. To effectively and efficiently optimize the high-level policy, we adopt a curriculum that gradually increases the difficulty of the task over time. In the first stage of the curriculum, the objective is to quickly explore the motion embedding and control the character to move in the right direction so that the racket gets close to the incoming ball. Therefore, we train the policy only with the racket position reward r_t^r, and use a larger learning rate ($1e^{-4}$), higher action distribution variance Σ_π (0.25), and a lower simulation frequency (120 Hz) for faster simulation. In the second stage of the curriculum, the goal is to control the racket so that the ball is hit over the net to the other side of the court. The target position is simplified as one of the three fixed positions at the left, center, and right of the court. In this stage, the policy is trained using both rewards with a higher weight on r_t^b (0.9), and using a smaller learning rate ($2e^{-5}$), a lower Σ_π (0.04). A higher simulation frequency (360 Hz) is also used by increasing the number of substeps in collision handling to ensure that racket-ball contact is simulated more accurately. Finally, the last stage of the curriculum encourages more precise control by sampling continuous target positions spanning the entire court. During this stage the policy is trained with an even smaller learning rate ($1e^{-5}$) and Σ_π (0.0025).

7.4 Additional Kinematic Constraints

In addition to the wrist motion, other aspects of routine tennis motion may not be reconstructed accurately from video data. Most notably, (1) the character may not consistently keep their eyes on the ball and (2) the character's non-dominant hand may not be gripping the racket during a two-handed swing.

While we could attempt to correct these errors by modifying the high-level policy to output corrections for more joints, and design

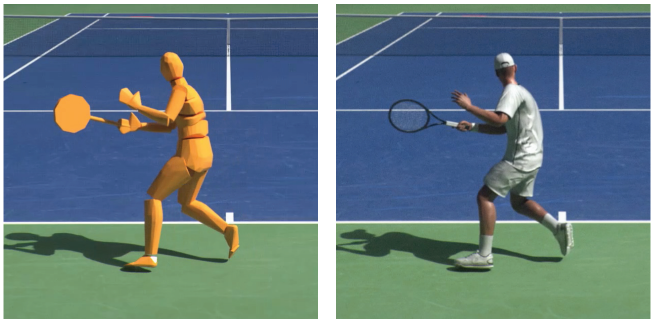

(a) Simulated Model (b) Visualization Model

Fig. 3. The simulated character model is created from SMPL [Loper et al. 2015], with 24 rigid body segments and 72 DOF. The tennis racket is a combination of two solid cylinders and the grip is simplified by directly attaching the end of the racket handle to the wrist joint.

specific rewards using domain knowledge of tennis, for simplicity we directly correct the kinematic motion with heuristics informed by knowledge of tennis.

Keeping eyes on the ball. In the real world, the player will rotate their head to keep their eyes on the ball. However, head/neck rotations are often poorly estimated by the kinematic pose estimator, leading to violations of this critical attribute of play. To correct head/neck motion, we first compute the offset angle between the head's facing direction and the direction from the head to the ball's current position and then add the offset angle back to the head/neck joints of the kinematic pose.

Keeping both hands on the racket for two-handed backhands. During a two-handed backhand swing, the player will hold the racket with both hands. However, pose estimation is typically not sufficiently accurate to yield both hands tightly gripping the racket. To improve the visual realism of the generated motion, we adjust the kinematic pose to move the free hand close to the racket handle by solving inverse kinematics for the joints from the wrist to the shoulder along the arm of the free hand.

We remind the reader that as with the output of the high-level policy, motion changes due to kinematic constraints are converted into physically plausible motions after imitation by the low-level policy.

8 PHYSICS MODELING OF TENNIS

In this section, we describe the modeling of the tennis racket and tennis ball used in our physics simulation.

Tennis racket and grip. The tennis racket is modeled as two solid cylinders with similar dimensions and masses as a real racket. The racket head is a rigid flat cylinder with a restitution of 0.9 and friction of 0.8 to simulate the effects of strings. We simplify the grip by directly attaching the end of the racket's handle to the character's wrist joint and model different grips by modifying the racket orientation relative to the character's palm (Figure 3). The relative orientation between the palm and the racket is set on a per-player basis according to the players' grip style ("eastern" grip for Federer's forehand, "semi-western" for Djokovic and Nadal).

Tennis ball. The tennis ball is simulated as a rigid sphere with the same radius and mass as a real tennis ball, with a restitution of 0.9 and friction of 0.8. To simulate air friction and the effects of spin, we add external air drag force F_d and Magnus force F_M into the simulation as follows:

$$F_d = C_d A v^2/2\,, \quad F_M = C_L A v^2/2\,, \tag{10}$$

where v denotes the magnitude of the ball's velocity and $A = \pi \rho R^2$ is a constant determined by the air density ρ and the ball's radius R. F_d is always opposite to the direction of the ball's velocity, and C_d refers to the air drag coefficient, which is set to a constant of 0.55. In tennis, topspin (forward ball rotation) imparts downward acceleration to the ball leading it to drop quickly. Backspin (backward ball rotation) produces upward acceleration causing the ball to float [Brody et al. 2004]. C_L refers to the lift coefficient due to the Magnus force and is computed as $1/(2 + v/v_{\text{spin}})$ where v_{spin} denotes the magnitude of ball's *spin velocity* (the relative speed of the surface of the ball compared to its center point). F_M is always perpendicular to the direction of the ball's angular velocity (following right-hand rule) and points downwards for topspin and upwards for backspin.

9 RESULTS AND EVALUATION

We evaluate our learned controllers quantitatively in terms of their ability to successfully perform tennis tasks and via metrics that model the quality of the resulting motion. We also qualitatively evaluate the ability of our learned controllers to produce tennis motions that are humanlike and also reflect aspects of the style of motion demonstrated in source videos. We encourage the reader to view the supplementary video for demonstrations of the motion generated by our system.

9.1 Experimental Setup

All physics simulations are implemented using Issac Gym [Makoviychuk et al. 2021]. All policies are implemented as neural networks using PyTorch and trained using Proximal Policy Optimization (PPO) [Schulman et al. 2017]. Advantage estimates for policy gradient updates are computed using the generalized advantage estimator GAE(λ) [Schulman et al. 2015], and gradient updates are performed using the Adam optimizer [Kingma and Ba 2014]. In our experiments, the low-level policy is trained with approximately one billion samples, which requires around 12 hours on a single NVIDIA RTX A100 GPU. Training the MVAE requires four hours and the high-level policy is trained over five billion samples, which requires about two days. Unless otherwise stated, results in this section have residual force control enabled. Full implementation details can be found in the supplementary material.

9.2 Metrics

To evaluate a simulated character's task performance and motion quality, we consider the following quantitative metrics:

Task performance metrics. The character is initialized to a position at the center of the baseline and tasked to hit 15 consecutive

Table 1. Task performance of controllers learned from three players' motions using our system. We show the 25%, 50%, and 75% quantiles using the metrics collected from 10K test sessions (15 consecutive balls per session). The learned controllers consistently hit a high fraction of balls back into the court, and achieve average bounce position errors of less than two meters.

	Hit rate	Bounce-in rate	Bounce-pos err (m)
Fed-full	0.85/0.92/1.00	0.77/0.85/0.92	1.49/1.74/1.93
Djo-full	0.92/0.92/1.00	0.73/0.81/0.85	1.16/1.37/1.68
Nad-full	0.92/0.92/1.00	0.69/0.77/0.85	1.31/1.56/1.89

random incoming tennis balls, which lasts about half a minute. The following statistics are then collected to evaluate the model's task performance:

- *Hit rate*: the fraction of shots where the racket contacts the incoming ball.
- *Bounce-in rate*: the fraction of shots where the ball is hit and it bounces inside the court on the opposite side.
- *Bounce position error (bounce-pos err)*: average distance between the target bounce position and a shot's actual bounce position when the ball lands inside the court.

Motion quality metrics. We also use the following metrics to measure the physical plausibility of the generated motions [Yi et al. 2021; Yuan et al. 2021]:

- *Jitter*: average of the third derivatives of all joint positions.
- *Foot sliding*: average displacement of body mesh vertices that contact the ground in two adjacent frames.

9.3 Learning Complex Tennis Skills

Using Federer's motion data, we train a single controller to move the simulated character to the incoming ball, perform the appropriate swing, and hit the ball to a target location with the desired spin (topspin or backspin). Figure 4 illustrates examples of the diversity of shots generated by the controller, which includes serves, topspin forehand shots, topspin backhand shots, and backhand slices. Note how the controller learns to perform the appropriate swing motion (Figure 4b: topspin backhand vs. 4d: slice backhand) to produce the target spin despite the lack of stroke annotations in the source video.

Task performance. To quantitatively evaluate task performance, we test the controller for 10K sessions (15 consecutive balls per session) and report statistics for all three task metrics in Table 1 (see row *Fed-full*). The controller is able to consistently hit incoming tennis balls despite their diverse trajectories (median hit rate: 0.92, median bounce-in rate: 0.85). The controller is also able to hit the ball near the specified target location. The median and mean bounce errors are both less than two meters.

Different player styles. One of the advantages of learning skills from large-scale video data is that our system can learn per-player motion embeddings from video clips of each player, and then train different high-level policies for each embedding. We demonstrate this by training controllers from video clips of Nadal and Djokovic in addition to Federer. The three players have distinct playing styles: Federer and Djokovic are both right-handed, but Nadal plays left-handed. Federer uses a one-handed backhand, while Djokovic and Nadal use two-handed backhands. Qualitatively, as shown in Figure 4, learned skills capture the coarse attributes of a player's style (handedness and whether they use a one or two-handed backhand). We report task performance for the controllers trained on Djokovic and Nadal videos in Table 1 (*Djo-full* and *Nad-full*). Our system learns high performing controllers in all three cases.

Table 2. Ablations on the effect of physics correction (PhysicsCorr) and hybrid control (HybridCtr). Removing either component of the system results in decreased task performance.

	Hit rate	Bounce-in rate	Bounce-pos err (m)
w/o PhysicsCorr	0.85/0.92/0.92	0.69/0.77/0.85	2.00/2.37/2.81
w/o HybridCtr	0.69/0.85/0.92	0.31/0.46/0.54	2.82/3.43/4.00
Fed-full	**0.85/0.92/1.00**	**0.77/0.85/0.92**	**1.49/1.74/1.93**

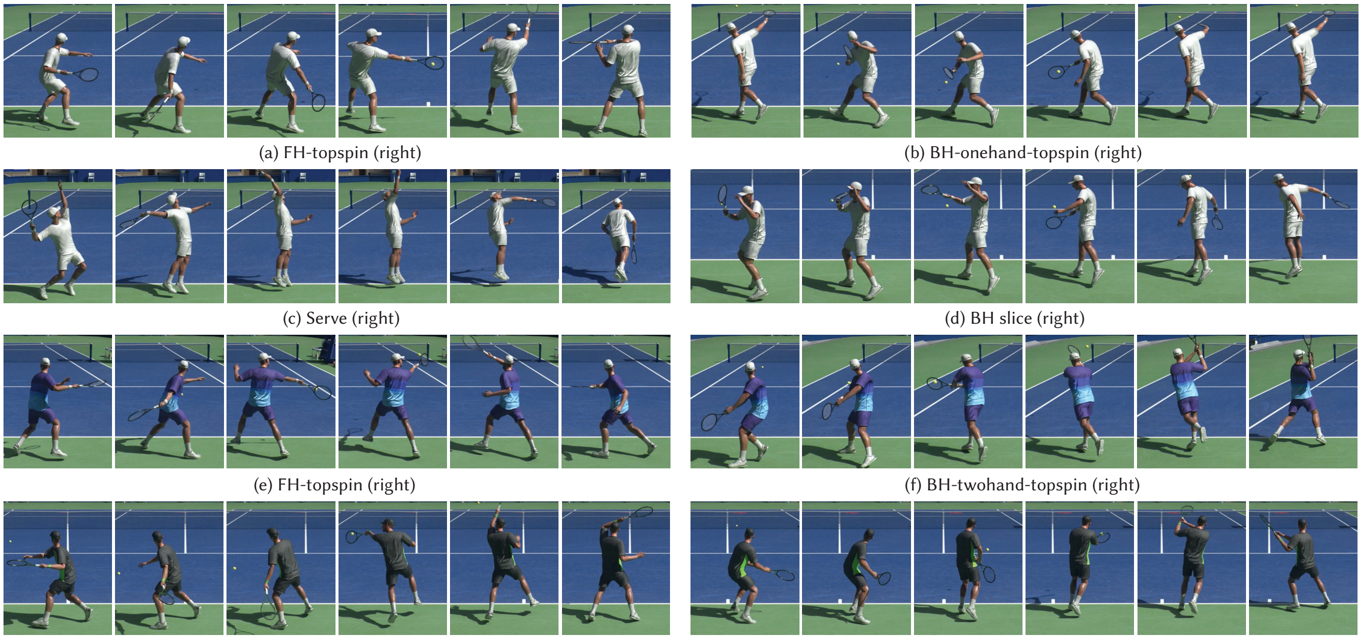
We note that although our per-player controllers successfully reflect gross aspects of each player's style, there remains much work to be done to accurately reproduce more nuanced details of professional-level tennis footwork and swings. For example, racket head velocity, and correspondingly the length of swing follow through, is shorter in the generated motions than the real-life examples. Wrist pronation, and thus racket head position during the back swing, is not faithfully reproduced. Also, our generated motions fail to capture how players place their non-swinging hand on the throat of the racket during swing preparation. To more accurately reproduce these details of professional athlete performance, higher-fidelity motion extraction from video and improved simulation fidelity (more accurate models of human anatomy and ball-string contact) are likely necessary.

Tennis rallies between two players. Although the controllers are trained in a single-player setting (i.e., a single simulated character without an opponent), once trained, they can be directly applied to a two-player simulation. Specifically, we use two trained controllers (of the same player or different players) to drive two simulated characters to play tennis rallies against each other. We were able to simulate a rally of 38 shots using the two controllers *Fed-full* and *Djo-full*, lasting for 41 seconds and ending with a miss by *Fed-full*.

9.4 Tackling Low-Quality Demonstrations

A key challenge in this work is learning from low-quality motion data extracted from videos. We conduct two ablation studies to show the effectiveness of our proposed solutions.

Constructing motion embedding. Our system leverages two key steps to process the noisy motions $\mathbb{M}_{kin}$ estimated by the kinematic pose estimator into smooth and plausible motions $\mathbb{M}_{vae}$. First, the noisy motions in $\mathbb{M}_{kin}$are corrected by the low-level imitation policy using physics simulation. Second, the corrected motions $\mathbb{M}_{corr}$ are further denoised by training MVAE to embed the motion into a smooth motion space. Table 3 shows the motion quality of the motions at different stages. The physics-corrected motion $\mathbb{M}_{corr}$ exhibits less jitter and foot sliding, and the motions generated by the MVAE ($\mathbb{M}_{vae}$) are even more smooth. As a side-effect, the smoothing



(a) FH-topspin (right)

(b) BH-onehand-topspin (right)

(c) Serve (right)

(d) BH slice (right)

(e) FH-topspin (right)

(f) BH-twohand-topspin (right)

(g) FH-topspin (left)

(h) BH-twohand-topspin (left)

Fig. 4. Our simulated characters demonstrate diverse tennis skills that reflect coarse characteristics of the per-player video data they were trained on (a)-(d) skills learned using Roger Federer's motion data, who is a right-handed player and uses one-handed backhand. (e)-(f) skills learned using Novak Djokovic's motion data, who is also a right-handed player but uses two-handed backhand. (g)-(h) skills learned using Rafael Nadal's motion data, who is a left-handed player and uses two-handed backhand.

Table 3. Motion quality evaluation. We compare the motion output by our full system (*Fed-full*), the motion from the ablation (*w/o PhysicsCorr*), and motions at different stages of our system: estimated kinematic motion ($\mathbb{M}_{kin}$), physically corrected motion ($\mathbb{M}_{corr}$), and motion output by MVAE ($\mathbb{M}_{vae}$). The motion generated from our full system shows higher motion quality (less jitter and foot sliding) than *w/o PhysicsCorr* and the motions at intermediate stages.

	Jitter (10^3 m/s^3)	Foot sliding (cm)
$\mathbb{M}_{kin}$	6.08	7.41
$\mathbb{M}_{corr}$	3.14	1.46
$\mathbb{M}_{vae}$	0.96	4.70
w/o PhysicsCorr	1.19	2.82
Fed-full	**0.51**	**1.46**

by MVAE also increases foot sliding in $\mathbb{M}_{vae}$, but these artifacts are largely removed by the final step of imitating $\mathbb{M}_{vae}$with physics simulation (*Fed-full*). To further evaluate the impact of the physics-based correction on the learned controllers, we can train the MVAE using the original outputs of the pose estimator $\mathbb{M}_{kin}$, and then use the resulting MVAE to train the high-level policy (*w/o PhysicsCorr*). Table 3 shows that the controller trained without physics-based correction produces motion with more jitter and foot sliding compared to *Fed-full.* Table 2 also shows that the task performance of the controller trained without correction also decreases, especially in bounce position error, which indicates the importance of using physics-based correction to construct a good motion embedding.

Hybrid control for wrist motion. To evaluate the effectiveness of the proposed hybrid control for the wrist motion, we can train the high-level policy to only predict the latent code for the MVAE and use the motion output from the MVAE without joint corrections as the target kinematic motion (*w/o HybridCtr*). As shown in Table 2, although the agent is still able to achieve a reasonable hit rate, the bounce-in rate drops nearly by half and bounce position error increases significantly. This indicates that the proposed hybrid control is essential for achieving the challenging task of returning the ball close to the target location.

9.5 Analysis of One Million Simulated Shots

To further understand the performance of our learned controllers, we simulate a million shots using *Fed-full* and analyze the performance metrics conditioned on certain features of the shots.

Incoming ball's velocity and spin. We plot the hit rate conditioned on the incoming ball's velocity and spin when launched, shown in Figure 5(a). Faster incoming balls are missed more often and balls with faster spin are also more difficult to hit, which is consistent with real-world tennis.

Incoming ball's bounce position. In Figure 5(b) and (c), we plot the hit rate and average bounce position error conditioned on the incoming ball's bounce position. We observe that balls that bounce

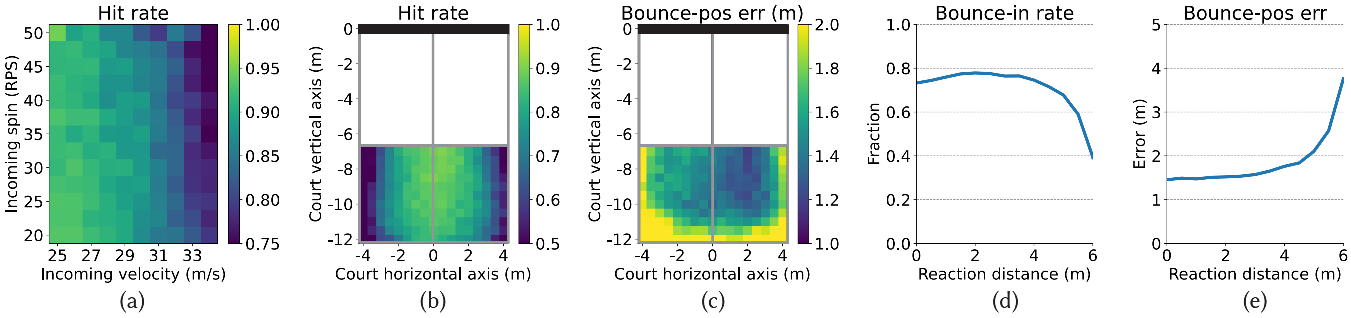


Fig. 5. Analysis of 1M simulated shots. (a) 2D heat maps of hit rate conditioned on incoming velocity and amount of spin. Balls with higher velocity and spin are harder to hit. (b)-(c) 2D heat maps of hit rate and average bounce position error conditioned on the incoming ball's bounce position. Balls that bounce closer to the edges of the court are harder to hit and result in higher bounce position errors. (d)-(e) Moving longer distances to reach an incoming ball (reaction distance) results in lower hit rates and larger bounce position errors since the character must move quickly and has less time to adjust.

Table 4. Ablation of various design choices of our system. The table provides average metrics collected from 10K test sessions. All design decisions contribute to the task performance of the controller.

	Hit rate	Bounce-in rate	Bounce-pos err (m)
w/o Root	0.61	0.45	2.28
w/o Phase	0.17	0.00	N/A
w/o FutureObs	0.86	0.30	4.51
w/o EstBounce	0.86	0.42	3.70
w/o Curriculum	0.89	0.71	2.67
Fed-full	**0.89**	**0.81**	**1.73**

shorter in the middle of the court are easier shots to hit, while balls that bounce close to the edges of the court are much more challenging (consistent with real tennis). A similar trend is also evident in Figure 5(c). Interestingly, incoming balls that bounce on the right side of the court are returned with lower bounce position error, suggesting that our right-handed simulated character has better control with forehand shots.

Reaction distance. Reaction distance is the distance that the character must move to reach an incoming ball. As shown in Figure 5(d) and (e), balls that require a longer reaction distance result in shots with a lower bounce-in rate and higher bounce position error. These balls are more difficult to reach and leave the character with less time to adjust.

9.6 Ablation Studies

We conduct additional ablations to isolate the importance of key design choices to the overall task performance. The results are summarized in Table 4. First, we observe that both adaptions to the original MVAE are crucial. Omitting the conditioning on global root position (*w/o Root*) results in global drifts of the kinematic motion due to error accumulation in autoregressive generation. The result is that the simulated character fails to hit subsequent shots. Further, without predicting the motion phase (*w/o Phase*), we are not able to minimize the distance between the racket and the ball at the ball contact time of the swing motion, i.e., the character can run into the ball at any time. The result is that the simulated character rarely hits the ball. Second, we find that it is important to estimate the ball trajectory in the future. In one ablation, we use historical ball positions instead of estimated future ball positions as the observation of the high-level policy (*w/o FutureObs*). Using historical ball positions results in a significant decrease in bounce-in rate, indicating the importance of estimated ball positions in the near future for more precise control, which is also a key skill for professional tennis players. In the other experiment, we compute the ball position reward (Equation 9) using the ball's current position in simulation instead of the estimated bounce position (*w/o EstBounce*). The significant decrease in the bounce-in rate and increase in bounce position error show that computing accurate, long-term reward using estimated bounce position is crucial for improving task performance. Finally, not adopting the proposed curriculum learning of the high-level policy (*w/o Curriculum*) also leads to inferior task performance, indicating that the proposed curriculum simplifies the task training.

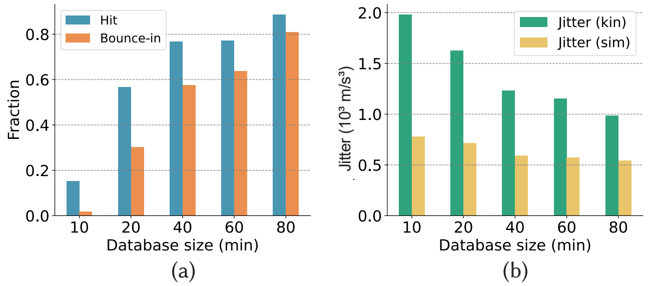


Fig. 6. Larger video database sizes (more motion) yield controllers with increased task performance (higher hit rate and bounce-in rate) and improved motion quality (lower jitter).

9.7 Sensitivity to Database Size

We also study the impact of input video database size on task performance and output motion quality (Figure 6). We retrain the motion embedding and the high-level policy using 12.5%, 25%, 50%, and

Table 5. Removing residual force control yields more realistic motion (40% reduction in foot sliding) but reduces the tracking ability of the low-level imitation policy, resulting in reduced overall task performance. Users can select whether or not to employ residual forces based on desired performance-motion quality needs.

	Hit rate	Bounce-in rate	Bounce-pos err (m)	Foot sliding (cm)
w/o RF	0.78	0.69	1.85	**0.87**
Fed-full	**0.89**	**0.81**	**1.73**	1.46

75% of the motion data for Federer. Figure 6(a) shows that using increasing motion data to construct the motion embedding improves task performance, suggesting the value of acquiring large amounts of motion data from large scale video datasets. With only 12.5% of data (about ten minutes of motion), the hit rate and bounce-in rate are significantly reduced because the learned motion embedding is not dense enough for successful high-level motion planning. Additionally, with less motion data available to model the latent space, the smoothing effect of the embedding is reduced due to decreased signal to noise ratio in reconstructing the repetitive motion, leading to more jitter in the decoded motion and the final simulated motion (Figure 6(b)).

9.8 Effects of Removing Residual Force Control

As described in Section 5, our system optionally allows for the development of controllers that use low-level imitation policies trained without residual force control. Table 5 evaluates the effect of removing residual force control on task performance and motion quality. Without residual force control, foot sliding is reduced by 40% and the resulting motions are perceptibly more plausible (see supplemental videos). However, the increased motion quality comes at the cost of a 12% reduction in hit rate and 15% decrease in bounce-in rate.

Our system provides users with the flexibility to choose the desired scale of residual forces based on their motion quality and task performance needs. Future work will explore ways to improve task performance without relying on residual force control.

10 DISCUSSION AND FUTURE WORK

Our system shows that widely available, but low-quality and largely unannotated demonstrations of human performance extracted from videos can be a rich data source for creating data-driven human animation controllers, provided task-specific dynamics simulation and deep reinforcement learning are used to correct for motion perception errors and learn high-level control policies.

We are able to produce physically simulated characters that succeed at the challenging task of playing extended tennis rallies, using a variety of shot types, while also reflecting coarse attributes of real human playing styles. However, as noted in Section 9.3, while the motion of our characters reflects human tennis play, there is still much to be done to faithfully reproduce the specific, fine-grained motion style of the professional athletes captured in the source videos. It will be interesting to consider how more detailed human muscular simulation (including fatigue) or ball-string contact modeling might lead to more accurate physically-based swing motions. It is also interesting to consider how access to a small number of high-quality demonstrations (via high-resolution video capture or mocap) could be combined with large scale, but low-quality performances extracted from broadcast videos to produce better output.

Our focus in this project was to successfully control characters to hit tennis balls in tennis rallies, not to learn strategies for winning tennis points (or full matches). It will be interesting to consider what strategies and styles of play emerge when characters are given the opportunity to make shot placement and court positioning decisions and trained at scale with competitive self play. Overall, we are excited about the possibilities of combining massive-scale human observations and simulated self-play to create human-like characters that learn to play complex athletic games in physically simulated environments.

ACKNOWLEDGMENTS

We would like to thank Tennis Channel[1] for providing the tennis video footage for this work. Additionally, we sincerely thank Steven Guevara, Alessandro Baldasseroni, Olivier Vernay Kim, Sergej Eichmann, Peter Wildman, EJ Holowicki, and Kevin Margo for creating the visualization for this work, and Steve Masseroni for narrating the supplementary video. Furthermore, we thank the anonymous reviewers for their feedback, James Hong, David Durst, Vishnu Sarukkai, and Purvi Goel for helpful discussions. This work was done while Haotian Zhang was an intern at Nvidia and it is partially supported by gifts from Meta.

REFERENCES

Anurag Arnab, Carl Doersch, and Andrew Zisserman. 2019. Exploiting temporal context for 3D human pose estimation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3395–3404.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems* 28 (2015).

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

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934* (2020).

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. 2016. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *European conference on computer vision*. Springer, 561–578.

H. Brody, R. Cross, and C. Lindsey. 2004. *The Physics and Technology of Tennis.* Racquet Tech Publishing.

Marcus A Brubaker, Leonid Sigal, and David J Fleet. 2009. Estimating contact dynamics. In *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2389–2396.

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

Stelian Coros, Philippe Beaudoin, and Michiel Van de Panne. 2010. Generalized biped walking control. *ACM Transactions On Graphics (TOG)* 29, 4 (2010), 1–9.

Rishabh Dabral, Anurag Mundhada, Uday Kusupati, Safeer Afaque, Abhishek Sharma, and Arjun Jain. 2018. Learning 3d human pose from structure and motion. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 668–683.

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

Dirk Farin, Susanne Krabbe, Wolfgang Effelsberg, et al. 2003. Robust camera calibration for sport videos using court models. In *Storage and Retrieval Methods and Applications for Multimedia 2004*, Vol. 5307. International Society for Optics and Photonics, 80–91.

Riza Alp Guler and Iasonas Kokkinos. 2019. Holopose: Holistic 3d human reconstruction in-the-wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10884–10894.

[1] www.tennischannel.com

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*. 71–78.

James Hong, Haotian Zhang, Michaël Gharbi, Matthew Fisher, and Kayvon Fatahalian. 2022. Spotting Temporally Precise, Fine-Grained Events in Video. In *European Conference on Computer Vision*. Springer, 33–51.

Seokpyo Hong, Daseong Han, Kyungmin Cho, Joseph S Shin, and Junyong Noh. 2019. Physics-based full-body soccer motion control for dribbling and shooting. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Mir Rayat Imtiaz Hossain and James J Little. 2018. Exploiting temporal information for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 68–84.

Yinghao Huang, Federica Bogo, Christoph Lassner, Angjoo Kanazawa, Peter V Gehler, Javier Romero, Ijaz Akhter, and Michael J Black. 2017. Towards accurate marker-less human shape and pose estimation over time. In *2017 international conference on 3D vision (3DV)*. IEEE, 421–430.

Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. 2018. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7122–7131.

Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. 2019. Learning 3d human dynamics from video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5614–5623.

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

Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. 2020. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5253–5263.

Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J Black, and Peter V Gehler. 2017. Unite the people: Closing the loop between 3d and 2d human representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6050–6059.

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-driven biped control. In *ACM SIGGRAPH 2010 papers*. 1–8.

Sergey Levine and Vladlen Koltun. 2013. Guided policy search. In *International conference on machine learning*. PMLR, 1–9.

Jiefeng Li, Chao Xu, Zhicun Chen, Siyuan Bian, Lixin Yang, and Cewu Lu. 2021. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3383–3393.

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

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

Libin Liu, Michiel Van De Panne, and KangKang Yin. 2016. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics (TOG)* 35, 3 (2016), 1–14.

Libin Liu, KangKang Yin, Michiel van de Panne, and Baining Guo. 2012. Terrain runner: control, parameterization, composition, and planning for highly dynamic motions. *ACM Trans. Graph.* 31, 6 (2012), 154–1.

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

Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, SM Eslami, Daniel Hennes, Wojciech M Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, et al. 2021. From motor control to team play in simulated humanoid football. *arXiv preprint arXiv:2105.12196* (2021).

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM transactions on graphics (TOG)* 34, 6 (2015), 1–16.

Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. 2019. AMASS: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5442–5451.

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

Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Srinath Sridhar, Gerard Pons-Moll, and Christian Theobalt. 2018. Single-shot multi-person 3d pose estimation from monocular rgb. In *2018 International Conference on 3D Vision (3DV)*. IEEE, 120–130.

Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. 2017. Vnect: Real-time 3d human pose estimation with a single rgb camera. *Acm transactions on graphics (tog)* 36, 4 (2017), 1–14.

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

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

Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter Gehler, and Bernt Schiele. 2018. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In *2018 international conference on 3D vision (3DV)*. IEEE, 484–494.

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

Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. 2018. Learning to estimate 3D human pose and shape from a single color image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 459–468.

Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 2019. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7753–7762.

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

Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Trans. Graph.* 36, 4, Article 41 (July 2017), 13 pages. 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, Article 94 (July 2022).

Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. 2018b. Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions On Graphics (TOG)* 37, 6 (2018), 1–14.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–20.

Marc H Raibert and Jessica K Hodgins. 1991. Animation of dynamic legged locomotion. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*. 349–358.

Davis Rempe, Leonidas J Guibas, Aaron Hertzmann, Bryan Russell, Ruben Villegas, and Jimei Yang. 2020. Contact and human dynamics from monocular video. In *European conference on computer vision*. Springer, 71–87.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

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

Soshi Shimada, Vladislav Golyanik, Weipeng Xu, Patrick Pérez, and Christian Theobalt. 2021. Neural monocular 3d human motion capture with physical awareness. *ACM Transactions on Graphics (ToG)* 40, 4 (2021), 1–15.

Soshi Shimada, Vladislav Golyanik, Weipeng Xu, and Christian Theobalt. 2020. Physcap: Physically plausible monocular 3d motion capture in real time. *ACM Transactions on Graphics (ToG)* 39, 6 (2020), 1–16.

Yu Sun, Yun Ye, Wu Liu, Wenpeng Gao, Yili Fu, and Tao Mei. 2019. Human mesh recovery from monocular images via a skeleton-disentangled representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5349–5358.

Jie Tan, Yuting Gu, C Karen Liu, and Greg Turk. 2014. Learning bicycle stunts. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–12.

Tianxin Tao, Matthew Wilson, Ruiyu Gou, and Michiel van de Panne. 2022. Learning to Get Up. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 47, 10 pages. https://doi.org/10.1145/3528233.3530697

TennisChannel. 2023. *Tennis Channel*. https://tennischannel.com

Hsiao-Yu Tung, Hsiao-Wei Tung, Ersin Yumer, and Katerina Fragkiadaki. 2017. Self-supervised learning of motion capture. *Advances in Neural Information Processing Systems* 30 (2017).

M Van de Panne and C Lee. 2003. Ski stunt simulator: Experiments with interactive dynamics. In *Proceedings of the 14th Western Computer Graphics Symposium*, Vol. 13. ACM Banff, AB, Canada.

Marek Vondrak, Leonid Sigal, Jessica Hodgins, and Odest Jenkins. 2012a. Video-based 3D motion capture through biped control. *ACM Transactions On Graphics (TOG)* 31, 4 (2012), 1–12.

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

Xiaolin Wei and Jinxiang Chai. 2010. Videomocap: Modeling physically realistic human motion from monocular video sequences. In *ACM SIGGRAPH 2010 papers*. 1–10.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2021. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–11.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2022. Physics-based character controllers using conditional VAEs. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–12.

Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. 2020. Allsteps: curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 213–224.

Zhaoming Xie, Sebastian Starke, Hung Yu Ling, and Michiel van de Panne. 2022. Learning Soccer Juggling Skills with Layer-wise Mixture-of-Experts. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.

Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. 2022. ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation. *arXiv preprint arXiv:2204.12484* (2022).

Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. 2022. ControlVAE: Model-Based Learning of Generative Controllers for Physics-Based Characters. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–16.

Xinyu Yi, Yuxiao Zhou, and Feng Xu. 2021. TransPose: real-time 3D human translation and pose estimation with six inertial sensors. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–13.

KangKang Yin, Stelian Coros, Philippe Beaudoin, and Michiel Van de Panne. 2008. Continuation methods for adapting simulated skills. In *ACM SIGGRAPH 2008 papers*. 1–7.

KangKang Yin, Kevin Loken, and Michiel Van de Panne. 2007. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 105–es.

Ri Yu, Hwangpil Park, and Jehee Lee. 2019. Figure skating simulation from video. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 225–234.

Ri Yu, Hwangpil Park, and Jehee Lee. 2021. Human dynamics from monocular video with dynamic camera movements. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–14.

Wenhao Yu, Greg Turk, and C Karen Liu. 2018. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Ye Yuan, Umar Iqbal, Pavlo Molchanov, Kris Kitani, and Jan Kautz. 2022. GLAMR: Global occlusion-aware human mesh recovery with dynamic cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11038–11049.

Ye Yuan and Kris Kitani. 2020. Residual force control for agile human behavior imitation and extended motion synthesis. *Advances in Neural Information Processing Systems* 33 (2020), 21763–21774.

Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. 2021. Simpoe: Simulated character control for 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7159–7169.

Petrissa Zell, Bastian Wandt, and Bodo Rosenhahn. 2017. Joint 3d human motion capture and physical analysis from monocular videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 17–26.

Haotian Zhang, Cristobal Sciutto, Maneesh Agrawala, and Kayvon Fatahalian. 2021. Vid2player: Controllable video sprites that behave and appear like professional tennis players. *ACM Transactions on Graphics (TOG)* 40, 3 (2021), 1–16.

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*. 89–96.