Learning Physically Simulated Tennis Skills from Broadcast Videos (supplementary material)

ACM Reference Format:
. 2023. Learning Physically Simulated Tennis Skills from Broadcast Videos (supplementary material). *ACM Trans. Graph.* 42, 4 (August 2023), 2 pages. https://doi.org/10.1145/3592408

1 VIDEO ANNOTATION

Here we list the 13 US Open matches we used to extract our tennis motion dataset.

- US Open 2019 Quarterfinal R. Federer vs. G. Dimitrov
- US Open 2019 Round of 16 R. Federer vs. D. Goffin
- US Open 2018 Round of 16 R. Federer vs. J. Millman
- US Open 2018 Third round R. Federer vs. N. Kyrgios
- US Open 2021 Final N. Djokovic vs. D. Medvedev
- US Open 2021 Semifinal N. Djokovic vs. A. Zverev
- US Open 2021 Quarterfinal N. Djokovic vs. M. Berrettini
- US Open 2021 Third round N. Djokovic vs. K. Nishikori
- US Open 2019 Final R. Nadal vs. D. Medvedev
- US Open 2019 Quarterfinal R. Nadal vs. D. Schwartzman
- US Open 2019 Round of 16 R. Nadal vs. M. Cilic
- US Open 2017 Final R. Nadal vs. K. Anderson
- US Open 2017 Semifinal R. Nadal vs. J. Del Potro

2 LOW-LEVEL IMITATION POLICY

Network. The policy is modeled by a neural network that maps a state $\mathbf{s}$ to a Gaussian distribution over actions $\pi(\mathbf{a}|\mathbf{s})$ with an input-dependent mean $\mu_\pi(\mathbf{s})$ and a fixed diagonal covariance matrix Σ_π. The mean is specified by a fully connected network with 3 hidden layers of [1024, 1024, 512] units, followed by linear output units. The value function $V(\mathbf{s})$ is modeled by a similar network, but with a single linear output unit. All the hidden units use ReLU activations [Nair and Hinton 2010].

Rewards. In all the experiments, we manually specify the weights and scales as follows: $\omega_o = 0.6$, $\omega_v = 0.1$, $\omega_p = 0.2$, $\omega_k = 0.1$, $\omega_e = 0.01$, $\alpha_o = 60$, $\alpha_v = 0.2$, $\alpha_p = 100$, $\alpha_k = 40$.

Training. The low-level policy is trained with 8,192 environments with a simulation frequency of 120 Hz. Hyper-parameters used during training of the low-level policy is available in Table 1. The low-level policy is first trained using AMASS dataset with about 1 billion samples. Next, the low-level policy is fine-tuned using the kinematic motion dataset ($\mathbb{M}_{kin}$) we extracted from tennis videos with about 1 billion samples, which can then be used to correct the kinematic motions and create the physically corrected motion dataset $\mathbb{M}_{corr}$. The low-level policy used in the control hierarchy for controlling the character's low-level movements is further fine-tuned using $\mathbb{M}_{corr}$ for each different player with about 0.25 billion samples.

Parameter	Value
Simulation frequency (Hz)	120
Σ_π Action distribution variance	0.03
Samples per update iteration	262144
Policy/value function minibatch Size	16384
γ Discount	0.99
Adam stepsize	0.00002
GAE(λ)	0.95
TD(λ)	0.95
PPO clip threshold	0.2
Episode length	300

Table 1. Hyper-parameters for training the low-level policy

To further remove the reliance of residual force control, we can keep fine-tune the policy by gradually reduce the maximum allowed residual forces/torques with about 1 billion samples, while the residual forces are reduced by 1% about every 10 million samples. We find it critical to slowly decrease the residual forces with sufficiently long training time.

3 MOTION EMBEDDING

Network. The encoder is a three-layer feed-forward neural network, with 256 hidden units in each internal layer followed by ELU activations [Clevert et al. 2015]. The output layer has two heads for μ and σ, required for the reparameterization trick used to train VAEs [Kingma and Welling 2013]. The decoder uses mixture-of-expert (MoE) architecture. Specifically, the MoE decoder consists of six identically structured expert networks and a single gating network to blend the weights of each expert to define the decoder network to be used at the current time step. Similar to the encoder, the gating network is also a three-layer feed-forward neural network with 256 hidden units followed by ELU activations. The input to the gating network is the latent variable z and the current pose. Each expert network is also similar to the encoder network in structure. These compute the next pose from the latent variable z and the current pose.

Training. Hyper-parameters used during training of the motion embedding model is available in Table 2. We adopt scheduled sampling [Bengio et al. 2015], where a sample probability p is defined for each epoch. The predicted pose is used as the input for the next time step with probability $1 - p$, otherwise, the ground-truth pose is used. The entire training process is divided into three modes:

Author's address:

0730-0301/2023/8-ART $15.00
https://doi.org/10.1145/3592408

Parameter	Value
Latent space dimension	32
Number of frames for condition	1
Number of frames for prediction	1
Sequence length	10
Number of seqs per epoch	50000
Batch size	100
Learning rate	0.0001

Table 2. Hyper-parameters for training the motion embedding model

supervised learning ($p = 1$), scheduled sampling (decaying p), and autoregressive prediction ($p = 0$). The number of epochs for each mode is 50, 50, and 400 respectively. For the scheduled sampling mode, the sampling probability decays to zero in a linear fashion with each learning iteration.

To train the model for predicting the motion phase with limited supervision (only 20% of the data is labeled with motion phase), we adopt a curriculum similar to scheduled sampling. We define a sample probability q, which specifies the probability of sampling a motion sequence labeled with motion phase. The entire training process is also divided into two stages: q decays linearly from 1 to 0.1 in the first stage, and stays at $q = 0.1$ for the second stage. Each stage is trained for 250 epochs.

4 HIGH-LEVEL MOTION PLANNING POLICY

Network. We adopt the same network architecture as the low-level policy.

Ball trajectory prediction model. The ball trajectory prediction model is used for estimating future incoming ball trajectory as the observation for the high-level policy, as well as estimating the outgoing ball bounce position for computing the ball position reward. At the offline stage, we compute a large ball trajectory pool by densely sampling the plausible ball states at launch time, including the height of the ball, the velocity of the ball and the spin velocity of the ball. The sample steps we used are 0.1m, 0.1m/s and 0.5 RPS, respectively. To reduce complexity, all the trajectories are computed in the Y-Z plane. The computed trajectories are stored in a dense matrix used as a lookup table. At runtime, a particular trajectory can be estimated by indexing the lookup table with the ball's launch state, rounded by the sample steps we used. In practice, we find this ball trajectory prediction model provides efficient and accurate estimates of the future ball positions and bounce position.

Rewards. In all the experiments, we manually specify the scales as follows: $\alpha_r = 0.05$, $\alpha_\theta = 0.1$.

Training. The high-level policy is trained with 30,720 environments. Hyper-parameters used during training of the high-level policy (with our proposed curriculum of three stages) is available in Table 3.

Parameter	Stage 1	Stage 2	Stage 3
Simulation frequency (Hz)	120	360	360
Σ_π Action distribution variance	0.25	0.04	0.0025
Samples per update iteration	327680	983040	983040
Policy/value function minibatch Size	16384	16384	16384
γ Discount	0.99	0.99	0.99
Adam stepsize	0.0001	0.00002	0.00001
GAE(λ)	0.95	0.95	0.95
TD(λ)	0.95	0.95	0.95
PPO clip threshold	0.2	0.2	0.2
Episode length	600	300	300

Table 3. Hyper-parameters for training the high-level policy with the curriculum of three stages.

REFERENCES

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).

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Icml*.