Learning Soccer Juggling Skills with Layer-wise Mixture-of-Experts

Zhaoming Xie
zxie47@cs.ubc.ca
University of British Columbia, Electronic Arts
Canada

Sebastian Starke
sstarke@ea.com
University of Edinburgh, Electronic Arts
United Kingdom

Hung Yu Ling
hyuling@cs.ubc.ca
University of British Columbia
Canada

Michiel van de Panne
van@cs.ubc.ca
University of British Columbia
Canada

Figure 1: Our system is able to generate many soccer juggling skills and their transitions. We show snapshots of a policy juggling with foot, chest and head.

ABSTRACT

Learning physics-based character controllers that can successfully integrate diverse motor skills using a single policy remains a challenging problem. We present a system to learn control policies for multiple soccer juggling skills, based on deep reinforcement learning. We introduce a task-description framework for these skills which facilitates the specification of individual soccer juggling tasks and the transitions between them. Desired motions can be authored using interpolation of crude reference poses or based on motion capture data. We show that a layer-wise mixture-of-experts architecture offers significant benefits. During training, transitions are chosen with the help of an adaptive random walk, in support of efficient learning. We demonstrate foot, head, knee, and chest juggles, foot stalls, the challenging around-the-world trick, as well as robust transitions. Our work provides a significant step towards realizing physics-based characters capable of the precision-based motor skills of human athletes. Code is available at https://github.com/ZhaomingXie/soccer_juggle_release.

CCS CONCEPTS

• **Computing methodologies** → *Physical simulation*; *Reinforcement learning*.

SIGGRAPH '22 Conference Proceedings, August 7–11, 2022, Vancouver, BC, Canada

ACM ISBN 978-1-4503-9337-9/22/08…$15.00
https://doi.org/10.1145/3528233.3530735

KEYWORDS

character animation, soccer juggling, reinforcement learning

ACM Reference Format:
Zhaoming Xie, Sebastian Starke, Hung Yu Ling, and Michiel van de Panne. 2022. Learning Soccer Juggling Skills with Layer-wise Mixture-of-Experts. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings (SIGGRAPH '22 Conference Proceedings), August 7–11, 2022, Vancouver, BC, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3528233.3530735

1 INTRODUCTION

Soccer is commonly acknowledged as being the most popular sport in the world. It is therefore also one of the most popular sports to be emulated for video games. The character animation required to play soccer is particularly challenging due to the complex interaction between the characters and the ball. In this paper, we tackle arguably some of the most complex interactions of this type, namely soccer juggling. This involves keeping the ball in the air via repeated hits using the feet, knees, head, or chest. Because of the challenging nature of these skills, soccer juggling is typically implemented using kinematic character animation, driven by motion capture data. As is common with this class of methods, however, this is then prone to a restricted degree of user control and blending artifacts.

In this paper, we present a system to generate soccer juggling animation using physics-based simulation and control. We first develop a control graph tailored to the soccer juggling problem. This control graph allows us to easily specify different soccer juggling skills via crude hand-designed pose sequences, or using motion capture data. Transitions between skills are introduced as directed edges in the graph. Reinforcement learning (RL) is used to train control policies based on this graph. In support of efficient and effective learning, we employ a layer-wise mixture-of-experts architecture,

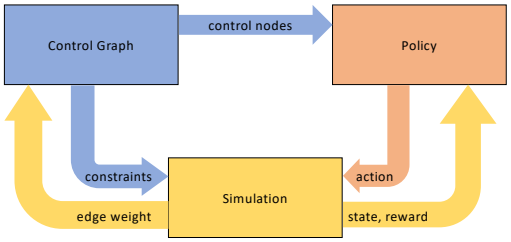

Figure 2: We design a control graph that specifies various juggling skills and their transitions. We learn these skills via random walk on the graph. The policy generates the action based on the upcoming control nodes and the simulation state. The policy is trained based on the reward feedback via RL. A simulation episode terminates if the constraints in the control node are violated, and the edge weight of the specific node will be updated to adjust the probability of traversing an edge during the random walk.

in contrast to the common output-gated mixture of experts. We further introduce success-adaptive random walks of the control graph for efficient and balanced training. The result is a control policy that can perform a variety of full-body soccer juggling skills and the related transitions, including foot, knee, head, and chest juggling, as well as the around the world foot juggle. See Fig. 1 for some examples.

This paper makes the following contributions:

- We propose an overall method for learning difficult soccer juggling skills. Individual skills can be defined using either a few hand-designed poses or using motion capture data.
- We show that a layer-wise mixture-of-experts architecture provides significant benefits for this multi-skill RL problem. Strong specialization among the experts tends to arise naturally, which reduces interference between different experts.
- We introduce an adaptive random walk training strategy in support of efficient learning.

2 RELATED WORK

In the interests of space, we largely focus on related work that uses physics-based simulations.

2.1 Physics-based Character Animation

A principled physics-based approach for modeling full-body motion has been a long-standing goal for creating realistic worlds, dating back over three decades, e.g., [van de Panne et al. 1990; Raibert and Hodgins 1991]. The development of control strategies, via careful design, optimization, and learning-based approaches, has been a key focus of efforts [Geijtenbeek and Pronost 2012]. The past decade has seen methods that leverage sampling-based methods [Hämäläinen et al. 2014, 2015; Liu et al. 2010, 2016], various types of policy search, e.g., [Geijtenbeek et al. 2013; Tan et al. 2014], reinforcement learning e.g., [Coros et al. 2009; Peng et al. 2015] and deep reinforcement learning, e.g., [Peng et al. 2016; Won et al. 2017; Heess et al. 2017].

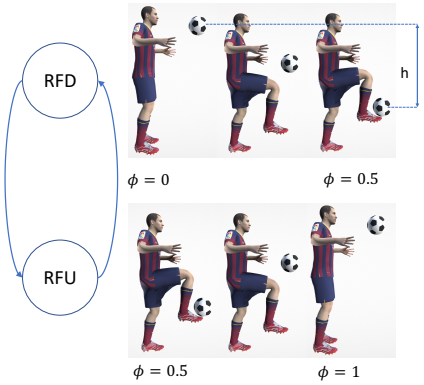

Figure 3: Stages of right foot juggling and the corresponding control nodes. A juggling skill is divided into two stages. Stage Down: the ball starts with zero velocity and falls down. Stage UP: the ball hits the foot and rises until the vertical velocity is 0. The phase ϕ goes from 0 to 0.5 in the down stage and goes from 0.5 to 1 in the up stage. The rate the phase changes depends on the overall duration of the stages, which is determined by the vertical traveling distance of the ball h. We build a control node for each stage and use directed edges to indicate transitions between stages.

Physics-based characters have been used to create athletic motions for various sports, including running, vaulting, and bicycling [Hodgins et al. 1995], via specialized control strategies. Recent examples include learning based approaches to generate skills such as bicycle stunts [Tan et al. 2014], wall climbing [Naderi et al. 2018], figure skating [Yu et al. 2019], parkour [Liu et al. 2012; Yu et al. 2021], jumping [Yin et al. 2021], boxing and fencing [Won et al. 2021].

2.2 Control while Interacting with Balls

Controlling a virtual character or robot to juggle a ball is a long-standing problem. A first scientific study of the ball juggling problem starts with [Shannon 1993]. Since then, there are many works that learn to juggle a ball using virtual hands, e.g., [Chemin and Lee 2018; Lee et al. 2018; Luo et al. 2021]. There are also works that learn to juggle with a bipededal robot [Poggensee et al. 2020], quadrotor [Müller et al. 2011; Dong et al. 2015] or robot arms [Serra et al. 2017]. These often assume full controllability of the robot.

Recent work in character animation also explores how to interact with objects while performing dynamic motions, such as basketball dribbling [Liu and Hodgins 2018; Park et al. 2019] and soccer dribbling [Peng et al. 2017, 2019; Liu et al. 2021]. These works often decouple the motion control and ball control, e.g., first learn to imitate a reference motion without the consideration of ball control and then learning to control the ball. [Hong et al. 2019] makes use of model predictive control to synthesize soccer dribbling skills, at the expense of slow online computation. More recently, [Peng et al. 2021] directly learn soccer dribbling skills from scratch with the guidance of a walking reference motion.

In this paper, we employ RL to synthesize control policies for soccer juggling skills. This requires the character to perform highly

dynamic skills while juggling a soccer ball using different body parts. Our system does not require a decoupling of learning the motion and controlling the ball, thus allowing for more efficient learning. Furthermore, with our fully trained policy, users can interactively choose what body parts to juggle the ball with at run time. This is in contrast with kinematics-based soccer juggling systems used in video games, where the user has limited control of the juggling sequence. [Jain and Liu 2009; Choi et al. 2015] also design motion synthesis systems to generate trajectories for soccer juggling. However, their system is designed for single trajectory generation and is thus not suitable for user interaction such as game control.

2.3 Mixture of Experts for Character Control

Mixture of experts (MOE), where multiple models are used to reconstruct the desired output, have been used extensively to generate complex motions for character animation and motion control.

Recent work in kinematics based character animation often makes use of the weight-blended mixture of experts neural network, where the weights of multiple feedforward neural networks are blended via a gating network to construct a single neural network. This has been used to generate complex kinematics behaviors via supervised learning, including human locomotion [Holden et al. 2017; Ling et al. 2020], quadruped locomotion [Zhang et al. 2018], humans interacting with objects [Starke et al. 2019], basketball skills [Starke et al. 2020] and boxing [Starke et al. 2021].

Recently, a weight-blended MOE has been used in RL for legged robot locomotion. However, it either shows no clear advantage over baseline methods for single task training [Yang et al. 2020a] or relies heavily on pretraining of individual skills for multitask training [Yang et al. 2020b]. In our work, we adopt a similar architecture, which we name a *layer-wise mixture of experts*, where the outputs of layers are blended instead of the weights. Note that since each layer is linear before the nonlinearity, this is equivalent to the weight-blended MOE, but the layer-wise blended interpretation affords a much more efficient implementation; see Appendix 1 for details. Our work is thus among the first to demonstrate the benefit of this class of architecture for RL settings.

The more conventional MOE architecture directly blends the final output of the individual expert neural nets. This enables policies to imitate a broad range of motion capture data [Won et al. 2020] with experts specialized in different motions, or generating distinct behaviors for different experts, which can then be used for policy transfer [Peng et al. 2019]. Instead of blending different experts to generate the output, other neural networks can be trained to select an appropriate expert to process the input. This has been used to learn locomotion policies for a 2D dog to traverse complex terrains [Peng et al. 2016]. All these methods rely on pretraining of different experts on specific variations of the final tasks.

3 SYSTEM OVERVIEW

We model the character with an articulated skeleton system, adapted from Deepmimic[Peng et al. 2018]. It has 34 degrees of freedom, including the floating base. We use RAISIM [Hwangbo et al. 2018] as the physics simulator. The character is controlled via Stable proportional-derivative controllers [Tan et al. 2011]. We simulate the soccer ball as a hollow sphere, with the dimensions and mass taken from a real soccer ball. We set the restitution coefficient between the soccer ball and the character to be 0.8.

An overview of our system is shown in Fig. 2. We use RL to train the character to juggle the ball. We simulate a large number of environments in parallel to collect data. This is made possible with the fast simulator, which can gather up to 800 seconds of simulated experience every second. In particular, we simulate 800 environments in parallel, with a 400 Hz simulation and controlled at 50 Hz. We use Proximal Policy Optimization [Schulman et al. 2017] as the RL algorithm. At each iteration, we collect 100 samples per environment for policy update, taking $2-3$ seconds on a machine with a NVIDIA GeForce RTX 3070 GPU and a 12-core CPU. Training takes 6 hours for single skill training and 30 hours for learning multiple skills and their transitions.

4 SOCCER JUGGLING WITH CONTROL GRAPHS

We design our control structure based in part on the desired trajectory of the ball and the body part that the ball should make contact with. In this section, we describe our control architecture that allows us to specify different juggling soccer skills as well as their transitions. There is some similarity between our control graph and motion graphs [Kovar et al. 2008], and related work that generates control policies with a motion graph, e.g., [Won et al. 2020].

4.1 Parameterization of a Soccer Juggling Skill

During juggling, the soccer ball switches between two stages: (1) falling downwards with gravity, until it hits a body part, after reached its peak height, and (2) moving upwards due to impact with the body parts and until reaching a maximum height. We further parameterize the ball trajectory and the character motion with a single phase variable $\phi \in [0, 1]$. Specifically, when the ball is at the peak with zero velocity, $\phi = 0$, ϕ advances linearly with time to $\frac{1}{2}$ as the ball goes down until the ball hits the desired body part, and linearly increases to 1 as ball moves up until the ball reaches 0 velocity at the peak. ϕ then resets to 0 and the whole process repeats. ϕ will also be used to compute the desired motion of the character so it is synchronized with the ball trajectory. Note that ϕ can increase at different rates for different skills or at different stages. At any stage of the juggling process, the ball will travel in the vertical direction for h meters, starting or ending with zero velocity. The overall duration of the stage can be computed as $t = \sqrt{2h/g}$ where g is the gravitational constant. The rate of change for ϕ is then $\sqrt{g/8h}$. In Fig. 3 we present an example of this parameterization.

4.2 Control Node

We design control nodes based on the stages of the ball, the body parts to receive the ball, and the overall motion of the character. A control node consists of the following:

- The phase speed, as calculated based on the overall motion duration, using the vertical traveling distance of the ball. The phase is integrated over time using the phase speed.
- The desired ball trajectory and the desired character motion.

- A constraint on when a body part can make contact with the ball.
- An optional constraint that enforces a relationship between the motion of the character and motion of the ball.
- A set of outgoing edges that indicate the possible transitions from the current control node to other nodes.

A control graph consists of a set of control nodes connected by a set of directed edges. A transition happens when the $\phi = \frac{1}{2}$ during a down stage or when $\phi = 1$ in an up stage. We next describe how to use RL to train control policies given a control graph and give examples of control graphs for soccer juggling skills.

4.3 Reinforcement Learning with A Control Graph

Given a control graph, we train policies to generate desired skills and their transition specified by the graph using RL. This is achieved via a random walk on the graph during training, and in each node, the policy is rewarded for completing the desired skills specified. Here, we document the details of the process.

Observation Space. The observation space for a control policy consists of the following:

- The state of the character, including the height, orientation and angular velocity of the root link in world space, the joint orientation and joint velocity of each joint. We exclude the root translation in the plane, as the policy should be invariant to that.
- The state of the ball, including its position relative to the character, and its angular velocity. Orientation is excluded.
- The phase and the phase speed.
- The current node the policy is in, and the next two nodes to be visited. This is encoded via three one-hot vectors, where the length of the vector is equal to the number of nodes. It is important for the policy to observe the upcoming nodes in order to make anticipatory decisions during transitions between many skills.

Action Space. We employ Proportional-Derivative (PD) controllers to control the character. Specifically, at each simulation time step, a control torque is computed via the following equation:

$$\tau = k_p(\theta_d \ominus \theta) - k_d\dot{\theta} + \tau_c, \tag{1}$$

where θ_d and θ is the desired and current joint orientation of the joints on the character, $\ominus$ computes the difference between two orientations, k_p and k_d are the control gains, and τ_c is a learned correction term to compensate for failure. We set θ_d to be the joint angle specified by the reference character pose in the current node given the phase, and the policy will output τ_c. For rotational joints, we can interpret τ_c as a correction term to the target angle, as has been done in prior works that use RL to track reference motions, e.g., [Xie et al. 2018; Park et al. 2019; Bergamin et al. 2019]. However, outputting a correction term for a spherical joint requires further transformations, such as taking the exponential map [Ma et al. 2021; Fussell et al. 2021]. We directly use the policy output as a torque correction term to avoid these complications.

Reward. We use a reward function of the form

$$r = r_{\text{character}} + r_{\text{ball}}, \tag{2}$$

Figure 4: Around the world motion sequence. The juggling foot will go around the ball. We put a coordinate system on the ball, with the origin at the center of the ball, the x-axis pointing in the lateral direction of the character and y-up. We can then describe the motion with ψ, which is the angle between the line connecting the ball and the foot and the x-axis. During training, ψ is enforced to stay within 60 degrees of ψ_d taken from the reference motion.

where $r_{\text{character}}$ is defined as

$$r_{\text{character}} = 0.4r_{\text{joint}} + 0.3r_{\text{translation}} + 0.3r_{\text{orientation}}, \tag{3}$$

and

$$\begin{aligned} r_{\text{joint}} &= \exp(-3\|(\theta \ominus \theta_d)\|^2) \\ r_{\text{translation}} &= \exp(-\|\dot{x} - \dot{x}_d\|^2 - \|\dot{y} - \dot{y}_d\|^2 - \|z - z_d\|^2) \\ r_{\text{orientation}} &= \exp(-2\|o \ominus o_d\|^2) \end{aligned}$$

rewards quantity in joint space, root-translational and rotational space to be close to the reference motion.

Lastly, r_{ball} is defined as

$$\begin{aligned} r_{\text{ball}} = \exp(&-\|x_{\text{ball, d}} - x_{\text{ball}} + x\|^2 \\ &- \|y_{\text{ball, d}} - y_{\text{ball}} + y\|^2 - 5\|z_{\text{ball, d}} - z_{\text{ball}}\|^2) \end{aligned}$$

where $x_{\text{ball, d}}$ and $y_{\text{ball, d}}$ is calculated based on the body part to juggle the ball, and $z_{\text{ball, d}}$ is the desired vertical ball location.

Initialization and Termination. Given a control graph, the policy randomly starts with a node corresponding to a down stage, with phase $\phi = 0$. The next two nodes are also sampled via random walk on the graph. The state of the character and the ball are initialized with the desired state specified by the reference motion. An episode terminates whenever the constraint specified in the node is violated, the orientation or height of the character reaches some threshold (a proxy falling condition) or when the ball hits the ground.

4.4 Single Skill Learning

A single juggling skill consists of the ball repeatedly going up and down, following a single desired trajectory, while a specific body part repeatedly hits the ball. The reference motion for the character is periodic. For convenience, we name a control node based on the up/down stage of the ball and the relevant body part. For example, the right foot juggling skill contains two nodes, named "Right Foot Down" (RFD) and "Right Foot Up" (RFU), respectively indicating that the ball is falling down and is about to be hit by the right foot, and that the ball is moving upward after being hit by the right foot.

4.4.1 Example 1: Juggling with the Right Foot with Hand Designed Reference. We start with a simple example of juggling with the right foot, described in Fig. 3. There are only two nodes in the graph, RFD and RFU, as described previously. The directed edges allow

us to generate a policy to transition between them. We design a simple reference motion with the character lifting the right foot up to meet the ball while the ball is falling (RFD) and putting the right foot down while the ball is rising (RFU). The other parts of the body remain stationary. More specifically, the motion of the right knee and the pitch angle of the right hip follows $\frac{\pi}{2}\sin(\phi/\pi)$. We further set a constraint where the ball is only allowed to make contact with the right foot or the right shin in the last 0.06s of RFD or the first 0.06s of RFU, and is not allowed to contact anything else. Despite the very crude reference motion for the character, our system is able to generate natural juggling skills.

4.4.2 Example 2: Juggling with the Right Foot with Around the World Reference. Around the world is a juggling trick where after kicking the ball with a foot, the same foot will go around the ball before kicking it again. We use motion capture data of this skill as our reference motion. The data is divided into two segments based on whether the ball is going up or going down. We then construct two control nodes for the around-the-world skill with directed edges, as shown in Fig. 4. To distinguish it from the normal right foot juggling, we name the control nodes "Right Around the World Down" and "Right Around the World Up", or "RAWD" and "RAWU".

One aspect of the around the world is the requirement for the foot to go around the ball. We find that using the reward alone is not sufficient to obtain the intended behavior. Inspired by spacetime bounds [Ma et al. 2021], in addition to the contact constraint similar to foot juggling, we introduce a constraint between the position of the foot and the ball, as shown in Fig. 4. Without this, the policy will cheat and juggle the ball in the normal fashion.

4.5 Transitions between Skills

Transitions between skills can be achieved by connecting nodes. Two nodes N_1 and N_2 can be connected if one of these two conditions apply: (i) the end pose of N_1 matches the starting pose of N_2, or (ii) the ball position at the end of N_1 relative to the body matches the ball position at the beginning of N_2.

4.5.1 Example 1: Transition between Foot Juggling and Around the World. First, we create a graph to describe the transition between foot juggling and around the world. In both foot juggling and around the world, the foot kicks the ball from beneath at the end of the down stage. At this point, the character can either put the foot down while the ball goes up or perform the around the ball skill by lifting the foot and moving it around the ball. This allows us to generate transition between foot juggling and around the world by connecting RFD with RAWU and RAWD with RFD. See Fig. 5(a).

4.5.2 Example 2: Transition between Foot Juggling and Head Juggling. Another common way to juggle a ball is using the head. We design a crude reference motion for head juggling where the character bends and straightens its knees during the down stage, in order to generate impulses to bounce the ball back up. During the up stage, the character assumes a standing pose. We name these nodes Head Down (HD) and Head Up (HU).

Since the right foot juggling skill and the head juggling skill have identical character poses at the beginning of the down stages, we can generate transitions between them by connecting RFU with HD and HU with RFD. Note that although the poses of the character

(a) Transition between right foot juggling and right around the world.

(b) Transition between right foot juggling and head juggling.

Figure 5: With our framework, we can define transitions between different skills by connecting the respective control nodes.

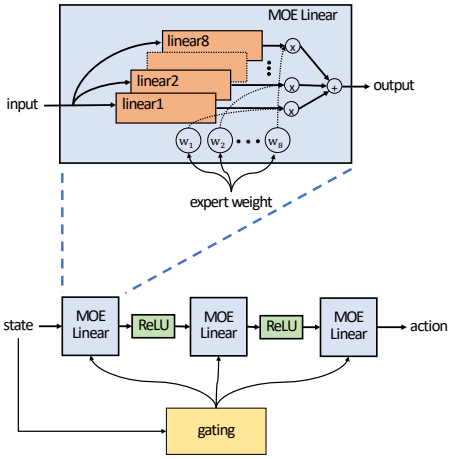

Figure 6: Layer-wise MOE: A linear MOE layer consists of multiple linear layers (experts) that are used independently to construct different outputs, these outputs are blended together via the expert weights.A layer-wise MOE consists of multiple layers of linear MOE, and a common gating network is used to generate the expert weights for all linear MOE layers.

are the same at the beginning, the positions of the ball are different, i.e., the ball is in front of the chest in RFD while the ball is on top of the head in HD. During RFU, depending on whether the next node is RFD or HD, the desired ball trajectories are different. Specifically, the traveling distance of the ball in the vertical direction is different, leading to slower phase speed if the next node is HD. See Fig. 5(b).

5 LEARNING MULTIPLE SKILLS AND TRANSITIONS

A soccer player is able to perform a wide variety of soccer juggling skills and can switch between them on the fly. To train policies with similar capability, we construct a large control graph that involves

skills such as juggling with different feet, different knees, head and juggling with different styles, i.e., around the world. The detailed graph is shown in Appendix 2. Policies are then trained via random walks on this graph. In this section, we describe key components that make training on this graph more efficient.

5.1 Layer-wise Mixture of Experts

While training for one or two skills is manageable with a standard multilayer perceptron (MLP) neural network, we find that as the number of skills increases, an MLP has a hard time mastering all the transitions. Mixture of experts (MOE), where multiple neural networks are used to construct the desired output, has been proven to be successful for multitask learning. We make use of a layer-wise MOE network, shown in Fig. 6, for more effective training. This is mathematically equivalent to the weight-blended MOE neural network, which has been used for synthesizing high quality kinematic based animation [Zhang et al. 2018; Starke et al. 2019, 2020, 2021; Ling et al. 2020]. We use a neural network with 3 linear MOE layers with 8 experts. Each expert is a linear layer with an output size 128 for the hidden linear MOE layer. The gating network is shared across all layers and is an MLP with 2 hidden layers of size 128. ReLU activations are used between different layers.

5.2 Adaptive Random Walk

Another challenge in RL with more skills is the data balancing problem. During random walk on the control graph, some transitions are inherently more challenging than others. During training, the easy transitions can be quickly learned before the policy can handle the more challenging transitions. Due to the termination of an episode after failure, more samples are accumulated for the easy transitions while fewer samples are obtained for harder ones. This unbalanced number of samples for different transitions makes the challenging transitions even harder to learn. To overcome this issue, we propose an adaptive weight update approach. We assign an integer count c to each edge in the graph, initialized to $c = 1$. During the random walk, when it is time to transition from one node to its neighbors, the probability of choosing which outgoing edge to use is proportional to the edge count c. Once an edge is selected, its count is incremented by 1. After transition to a new node, if the policy is able to complete the motion of the node, i.e., no termination condition is triggered, then the edge count is decreased by 2. The edge weights are constrained to $c \in [1, 100]$ to avoid oversampling an edge or starving edges of visits. Under this setup, a transition is more likely to happen if this transition often leads to failure, while the probability of successful transitions being selected will decrease. Similar ideas are explored in [Won and Lee 2019], where the value function is used to estimate task difficulty.

6 RESULTS

Our system is able to train policies for a character to juggle the soccer with different body parts. We encourage the readers to watch the supplementary video for qualitative results. In this section, we evaluate the importance of different components of our system.

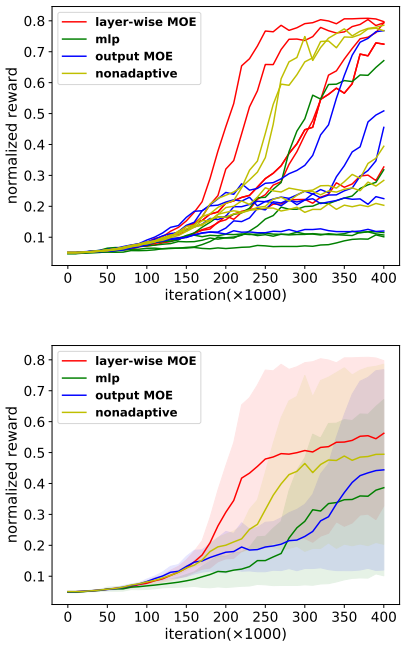

Figure 7: Learning curves over 5 runs with different settings. TOP: Learning curves for individual runs. BOTTOM: Average performance of different runs.

6.1 Evaluation of Layer-wise Mixture of Experts

We evaluate the importance of using a layer-wise MOE for training on a control graph with many skills and transitions. Specifically, we compare layer-wise MOE with a MLP and a standard MOE where only the outputs are blended, which has been used by many prior works in RL for character animation [Won et al. 2020; Peng et al. 2019]. For fair comparison, we use the same number of neurons for MLP and the layer-wise MOE. Specifically, each hidden layer of MLP has $8 \times 128 = 1024$ neurons. This introduces more parameters for the MLP compared to layer-wise MOE since the MLP is fully connected. Similarly, we use an output MOE that has the same number of parameters and internal structure as the layer-wise MOE. The learning curves of 5 runs using different architectures are shown in Fig 7. Overall, the layer-wise MOE learns significantly faster.

We hypothesize that the benefit of using layer-wise MOE is its capability to learn different skills using a different set of experts, reducing the interference effect that can be present in a fully connected MLP [Yu et al. 2020]. In Fig. 8, we show the gating patterns for different skills of a layer-wise MOE policy. We also collect the output of the second layer and the final output of different experts over 200 timesteps and project them in 2D with t-SNE. The distinct clusters demonstrate that the neural network does not converge to a scenario where all experts are learning the same distribution.

We introduce two metrics to measure the performance of a MOE, named specialization and utilization. If we run an episode for N

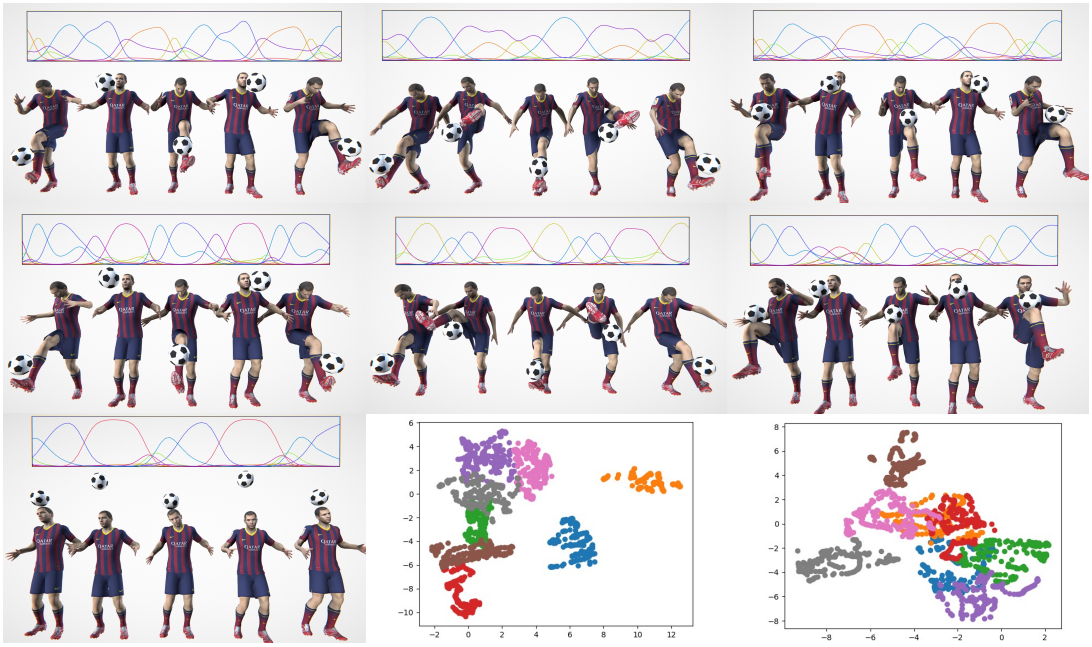

Figure 8: Different skills exhibit different gating patterns. We also visualize the output space of different layers of different experts.

Table 1: We apply a constant force to the soccer ball in different directions and record the maximum force different skills can keep juggling with. For reference, a constant force of 1 Newton applied on the ball will be approximately equivalent to a wind of speed 24 km/h acting on the ball.

	forward	backward	left	right
LF	0.45	0.23	0.34	0.34
LK	0.27	0.34	0.47	0.34
LAW	0.45	0.90	0.56	0.56
H	0.09	0.45	0.27	0.18
RF	0.02	0.23	0.18	0.40
RK	0.27	0.41	0.17	0.34
RAW	0.27	0.90	0.56	0.61

steps, the specialization and utilization are computed as

$$spec = \frac{1}{N}\sum_{i=1}^{N}(\sum_{k=1}^{K} g_{i,k}^2), util = \sum_{k=1}^{K}(\frac{1}{N}\sum_{i=1}^{N} g_{i,k})^2, \quad (4)$$

where K is the number of experts and $g_{i,k}$ is the gating weight for expert k at time step i. Intuitively, specialization measures the average activation pattern of the experts at each time step, while utilization measures whether the policy utilizes all the experts. In our scenario with 8 experts, extreme-but-balanced specialization would have only 1 expert at any point in time, with all experts used equally often. In this case, the optimal specialization value is 1 (higher is better) while the optimal utilization value is 0.125 (lower is better). We measure the specialization and utilization for layer-wise MOE and output MOE, via 1 minute of random walk on the control graph. For layer-wise MOE, we see a specialization of 0.664 ± 0.050 and utilization of 0.181 ± 0.030, averaged over the final policies of 5 different runs. For output MOE, we see a specialization of 0.394 ± 0.013 and a utilization of 0.136 ± 0.008. We see that layer-wise MOE induces better specialization, which can reduce the interference effect between tasks. The slightly worse utilization is expected since the control graph is unbalanced.

6.2 Ablations on Methodology

Adaptive Random Walk. The adaptive random walk supports the learning of challenging transitions. We run an ablation where all transitions are equally sampled, with results shown in Fig. 7. Note that the same initialization of the policies is used for fairness. In all cases, training with adaptive random walk converges faster. We note that even without the adaptive random walk, the layer-wise MOE is better than alternatives with the adaptive random walk, further demonstrating the benefit of using the layer-wise MOE.

Constraints Enforcement. If we remove the contact constraint for a skill, a policy fails to juggle with the desired body parts and does not respect contact timing. For the around-the-world, the constraint on the relative position between the foot and the ball is also crucial to generate the desired behavior. Please see video for the results.

6.3 Robustness

We evaluate the robustness of a control policy by adding external forces to the soccer ball in different directions. The results are recorded in Table 1. Our learned policy can withstand perturbations equivalent to a moderate breeze. We also evaluate the robustness of the policy against object shapes. Surprisingly, we find that our policy is able to juggle novel shapes such as box, cylinder and ellipsoid, with sizes similar to that of the soccer ball.

We can also generate turning motions by slowly rotating the appropriate observations, such as the orientation of the character. Even though the policy is not trained with turning motions, it observes similar states during training and is able to turn.

7 CONCLUSIONS

We present a system for learning soccer juggling skills with a physically simulated character. We design a control architecture that allows us to design and train different juggling skills, such as juggling with different body parts and juggling with different styles such as the challenging around the world skill. Transitions between different skills can also be easily defined with our control architecture. We further demonstrate efficient training utilizing adaptive random walks and the layer-wise MOE.

Currently we need to manually design control nodes for each new skills. An automatic way to create control nodes and attach them to existing control graphs can make learning more skills scalable. This may be achieved via automatic synthesis from juggling motion capture data or videos. It will be also interesting to synthesize novel skills automatically, e.g., by rewarding policies for generating novel motion sequences.

While we design our control graph for solving the soccer juggling problem, we believe it can be extended to similar control problems that involve interaction between the characters and objects, e.g., basketball dribbling. We can also extend this architecture to model interactions between multiple characters, e.g., passing the soccer ball between them while juggling.

The layer-wise MOE architecture demonstrates better performance compared to other alternatives. We hypothesize that it is because it reduces the interference effect in the multi-task setting. This is shown via the specialization metric we introduce. In the future, we plan to investigate more design alternatives in order to understand and further improve this architecture and apply it to more general muiltitask settings.

ACKNOWLEDGMENTS

We thank Yiwei Zhao for preparing the motion capture data for around the world juggling and helpful discussion. We also thank Harold Chaput for his support of the project.

REFERENCES

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

Jason Chemin and Jehee Lee. 2018. A physics-based juggling simulation using reinforcement learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*. 1–7.

Jong-In Choi, Shin-Jin Kang, Chang-Hun Kim, and Jung Lee. 2015. Virtual ball player. *The Visual Computer* 31, 6 (2015), 905–914.

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2009. Robust task-based control policies for physics-based characters. In *ACM SIGGRAPH Asia 2009 papers*. 1–9.

Wei Dong, Guo-Ying Gu, Ye Ding, Xiangyang Zhu, and Han Ding. 2015. Ball juggling with an under-actuated flying robot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 68–73.

Levi Fussell, Kevin Bergamin, and Daniel Holden. 2021. SuperTrack: motion tracking for physically simulated characters using supervised learning. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–13.

Thomas Geijtenbeek and Nicolas Pronost. 2012. Interactive character animation using simulated physics: A state-of-the-art review. In *Computer graphics forum*, Vol. 31. Wiley Online Library, 2492–2515.

Thomas Geijtenbeek, Michiel van de Panne, and A Frank Van Der Stappen. 2013. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–11.

Perttu Hämäläinen, Sebastian Eriksson, Esa Tanskanen, Ville Kyrki, and Jaakko Lehtinen. 2014. Online motion synthesis using sequential monte carlo. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–12.

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

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. 2017. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286* (2017).

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.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

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.

Jemin Hwangbo, Joonho Lee, and Marco Hutter. 2018. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters* 3, 2 (2018), 895–902. www.raisim.com

Sumit Jain and C Karen Liu. 2009. Interactive synthesis of human-object interaction. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 47–53.

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2008. Motion graphs. In *ACM SIGGRAPH 2008 classes*. 1–10.

Seunghwan Lee, Ri Yu, Jungnam Park, Mridul Aanjaneya, Eftychios Sifakis, and Jehee Lee. 2018. Dexterous manipulation and control with volumetric muscles. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

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 Transactions on Graphics (TOG)* 31, 6 (2012), 154.

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).

Yunhao Luo, Kaixiang Xie, Sheldon Andrews, and Paul Kry. 2021. Catching and Throwing Control of a Physically Simulated Hand. In *Motion, Interaction and Games*. 1–7.

Li-Ke Ma, Zeshi Yang, Xin Tong, Baining Guo, and KangKang Yin. 2021. Learning and Exploring Motor Skills with Spacetime Bounds. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 251–263.

Mark Müller, Sergei Lupashin, and Raffaello D'Andrea. 2011. Quadrocopter ball juggling. In *2011 IEEE/RSJ international conference on Intelligent Robots and Systems*. IEEE, 5113–5120.

Kourosh Naderi, Amin Babadi, and Perttu Hämäläinen. 2018. Learning physically based humanoid climbing movements. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 69–80.

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

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

Xue Bin Peng, Glen Berseth, and Michiel van de Panne. 2015. Dynamic terrain traversal skills using reinforcement learning. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.

Xue Bin Peng, Glen Berseth, and Michiel van de Panne. 2016. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.

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

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

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control. *ACM Trans. Graph.* 40, 4, Article 1 (July 2021), 15 pages. https://doi.org/10.1145/3450626.3459670

Katherine L Poggensee, Albert H Li, Daniel Sotsaikich, Bike Zhang, Prasanth Kotaru, Mark Mueller, and Koushil Sreenath. 2020. Ball Juggling on the Bipedal Robot Cassie. In *2020 European Control Conference (ECC)*. IEEE, 875–880.

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.

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

Diana Serra, Fabio Ruggiero, Vincenzo Lippiello, and Bruno Siciliano. 2017. A nonlinear least squares approach for nonprehensile dual-hand robotic ball juggling. *IFAC-PapersOnLine* 50, 1 (2017), 11485–11490.

Claude E Shannon. 1993. Scientific aspects of juggling. , 924 pages.

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural state machine for character-scene interactions. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.

Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. 2020. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 54–1.

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

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

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

Michiel van de Panne, Eugene Fiume, and Zvonko Vranesic. 1990. Reusable motion synthesis using state-space controllers. *ACM SIGGRAPH Computer Graphics* 24, 4 (1990), 225–234.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 33–1.

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

Jungdam Won and Jehee Lee. 2019. Learning body shape variation in physics-based characters. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12.

Jungdam Won, Jongho Park, Kwanyu Kim, and Jehee Lee. 2017. How to train your dragon: example-guided control of flapping flight. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–13.

Zhaoming Xie, Glen Berseth, Patrick Clary, Jonathan Hurst, and Michiel van de Panne. 2018. Feedback control for cassie with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1241–1246.

Chuanyu Yang, Kai Yuan, Shuai Heng, Taku Komura, and Zhibin Li. 2020a. Learning natural locomotion behaviors for humanoid robots using human bias. *IEEE Robotics and Automation Letters* 5, 2 (2020), 2610–2617.

Chuanyu Yang, Kai Yuan, Qiuguo Zhu, Wanming Yu, and Zhibin Li. 2020b. Multi-expert learning of adaptive legged locomotion. *Science Robotics* 5, 49 (2020).

Zhiqi Yin, Zeshi Yang, Michiel van de Panne, and KangKang Yin. 2021. Discovering diverse athletic jumping strategies. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–17.

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.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient Surgery for Multi-Task Learning. *Advances in Neural Information Processing Systems* 33 (2020).

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.

A WEIGHT-BLENDED MOE VS LAYER-WISED MOE

A.1 Equivalency between Weight-blended MOE and Layer-wise MOE

The forward pass of a fully connected neural network with L layers involves the following computations for each layer l:

$$x_l = activation(W_l x_{l-1}),$$

where activation is a fixed nonlinear function, x_0 is the input, W_l is the learnable matrix at layer l. Note that we omit the bias term but it can be included by appending a 1 to all x_l. For a weight-blended MOE, the computation at each layer will be:

$$x_l = activation(\sum_g (w_g W_{g,l}) x_{l-1}),$$

where w_g and $W_{g,l}$ is the gating weight and the corresponding expert parameters for expert g. Since every operation before the activation is linear, this is equivalent to

$$x_l = activation(\sum_g w_g (W_{g,l} x_{l-1})),$$

where instead of computing the weighted sum of $W_{g,l}$ first, we compute $W_{g,l} x_{l-1}$ independently first and compute the weighted sum of their results. This corresponds to our formulation of layer-wise MOE.

A.2 Computational Efficiency

Even though the weight-blended interpretation and layer-wise MOE interpretation are equivalent, they require dramatically different compute resources. In the weight-blended interpretation, when we desire to perform forward pass or backward pass with a batch of data with batch size B, we need to copy all the parameters in $W_{g,l}$ B times in order to perform batch operation. This requires additional memory of $O(Bmn)$ where m, n are the size of the matrix $W_{g,l}$. On the other hand, layer-wise MOE can perform this operation in the usual manner without additional copying of data. The speed up of layer-wise MOE compared to weight-blended MOE is related to the batch size B. When B is small (around 32-64) as in the supervised learning setting, the speedup is around 10x. The speedup grows almost linearly with the batch size. Weight-blended MOE used up all $8GB$ of GPU memory when the batch size reaches 512 and stops working, while layer-wise MOE continues to operate efficiently. In RL, we use large batch size (10000), which is impossible to implement using weight-blended MOE with our hardware setup. Our layer-wise MOE is inspired by the implementation from [Ling et al. 2020]; however, we focus on physics-based motion control, while theirs is kinematic.

B ADJACENCY MATRIX FOR MULTIPLE SKILLS AND THEIR TRANSITIONS

We design many soccer juggling skills and their transitions with our control graph. In Table 2, we show the adjacency matrix for this control graph.

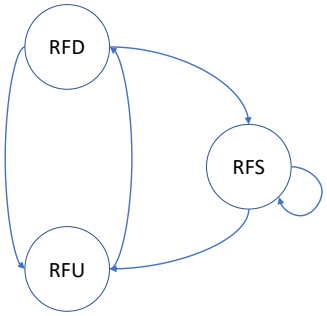

Figure 9: The control graph for creating a foot stall policy.

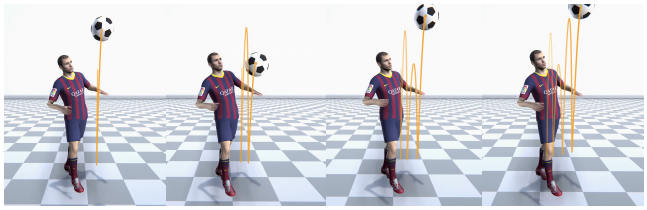

Figure 10: We can generate a foot juggling policy that juggles the ball to different height.

C ADDITIONAL RESULTS

C.1 Chest Juggling

We also create a chest juggling skill, where the chest of the character is used to receive and bounce the ball. See Fig. 1 for visualization.

C.2 Foot Stalling

Another common soccer juggling skill is foot stalling. To create a foot stalling skill, we create an additional control node called Right Foot Stall, or RFS. In the right foot stall, the reference motion is a fixed pose with the character standing on the left leg with the right leg lifted up, and the soccer ball lying still on top of the right foot. Directed edges from RFD to RFS and RFS to RFU are created to indicate transitions between them. To generate foot stalling motion with appropriate length, another edge connecting RFS to itself is also created. The weight of this self connected edge is set to be 30 to indicate a desired longer duration of the stalling motion. See Fig. 9 for the control graph for generating a foot stall policy.

C.3 Foot Juggling with Different Heights

For the foot juggling skill, we can randomize the desired traveling distance of ball h in the vertical direction to train policy that can be capable of juggling the soccer ball at different heights. See Fig. 10 for visualization.

Table 2: Adjacency matrix for control graph with many skills. 1 **indicates a directed edge going from the row node to the column node.**

	LFD	LFU	LKD	LKU	LAWD	LAWU	HD	HU	RFD	RFU	RKD	RKU	RAWD	RAWU
LFD		1				1								
LFU	1		1				1		1		1			
LKD				1										
LKU	1		1				1		1		1			
LAWD		1				1								
LAWU					1									
HD								1						
HU	1		1				1		1		1			
RFD										1				1
RFU	1		1				1		1		1			
RKD												1		
RKU	1		1				1		1		1			
RAWD										1				1
RAWU													1	