Robotic Table Tennis with Model-Free Reinforcement Learning

Wenbo Gao*§, Laura Graesser*†, Krzysztof Choromanski*†, Xingyou Song†, Nevena Lazic‡,
Pannag Sanketi†, Vikas Sindhwani† and Navdeep Jaitly¶
*Equal contribution †Robotics at Google ‡DeepMind
§Columbia University. Work done during Google internship ¶Work done at Google

***Abstract*—We propose a model-free algorithm for learning efficient policies capable of returning table tennis balls by controlling robot joints at a rate of 100Hz. We demonstrate that evolutionary search (ES) methods acting on CNN-based policy architectures for non-visual inputs and convolving across time learn compact controllers leading to smooth motions. Furthermore, we show that with appropriately tuned curriculum learning on the task and rewards, policies are capable of developing multi-modal styles, specifically forehand and backhand stroke, whilst achieving 80% return rate on a wide range of ball throws. We observe that multi-modality does not require any architectural priors, such as multi-head architectures or hierarchical policies.**

I. Introduction

Recent advances in machine learning (ML), and in particular *reinforcement learning* (RL), have led to progress in robotics [17, 43, 16], which has traditionally focused on optimal control. Two key advantages of ML are its ability to *leverage increasing data and computation* and *learn task-specific representations*. ML algorithms reduce the need for human knowledge by automatically learning useful representations of the data. For difficult problems, this is crucial because the complexity often exceeds what can be accomplished by explicit engineering. The effectiveness of ML is dependent on having large amounts of data. While this limits ML when little data is available, it becomes a strength because ML *continues* to scale with ever-increasing amounts of data and computation: a system can be improved, often to the new state of the art, simply by gathering more data. These advantages make RL appealing for robotics, where large-scale data can be generated by simulation, and systems and tasks are often too complex to explicitly program.

In this paper, we apply RL to robotic table tennis. In contrast to some of the previous manipulation tasks solved by RL [17] where inference time of few hundred milliseconds is acceptable, table tennis is distinguished by the need for *high-speed* perception and control, and the high degree of precision required to succeed at the task — the ball needs to be hit very precisely in time and space. Given these challenges, prior work on robotic table tennis is typically model-based, and combines kinematics for predicting the full ball trajectory with inverse kinematics for planning the robot trajectory. Most recent research first identifies a *virtual hitting point* [35] from a partial ball trajectory and predicts the ball velocity, and potentially spin, at this point. A target paddle orientation and velocity is then determined, and a trajectory is generated to bring the paddle to the desired target at a particular time [26, 27, 31, 29, 24, 13, 21]. Most systems include a predictive model of the ball (learned or analytical), and may also utilize a model of the robot. It is an open question whether end-to-end RL is an effective approach for robotic table tennis (and complex high-speed tasks in general), which motivates our approach.

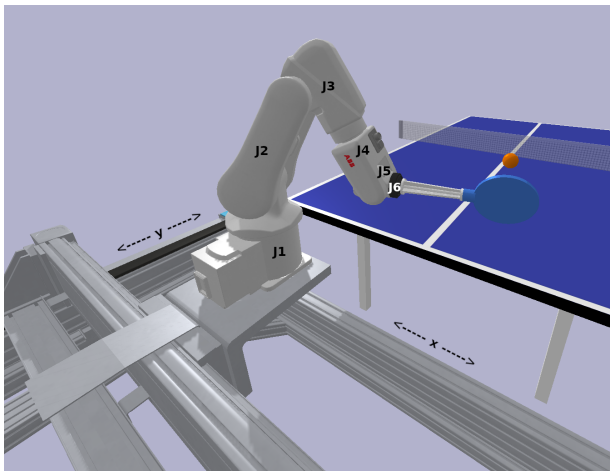

Fig. 1. Simulated table tennis 8 DOF robot: 6 DOF arm with revolute joints (labeled j1-j6) + 2 linear axes.

We are also motivated by human play. When humans play table tennis, they exhibit a variety of stroke styles (multi-modality). These styles include forehand, backhand, topspin, backspin, etc. We are interested in understanding if multi-modal style emerges naturally during training, and if not, what techniques are required to generate this behavior.

In this work, we describe how to learn a multi-modal model-free policy to directly control a simulated table tennis robot in joint space using RL, without relying on human demonstrations, and without having a separate system to predict the ball trajectory. To simplify the task, we focus on forehand and backhand play styles without spin. Our policies take as input short histories of joint positions and ball locations, and output velocities per joint at 100Hz.

A video demonstrating our system can be viewed at www.youtube.com/watch?v=-eHeq1nvHAE. The video is di-

vided into segments, each exhibiting a different policy (*Policy A – E*). Policy A is an example of a strong policy which returns **80%** of randomly thrown balls to the opponent side of the table (see Table VII), and exhibits high-level decisions through *bimodal* (forehand and backhand) play. Policies B – E in the video show ablation studies (Section V-C to V-E) and failure modes (Section V-F2). Policies B and C illustrate the differences in style between CNN and MLP architectures.

In summary, our contributions are the following:

- To the best of our knowledge, we train the first table tennis controller using model-free techniques without a predictive model of the ball or human demonstration data (Section V-B).
- We show that it is possible for a policy to learn multi-modal play styles with careful curriculum learning, but with no need of architectural priors (Section V-F).
- We demonstrate that convolutions across time on non-visual inputs lead to smoother policies. They are also more compact than MLPs, which may be particularly beneficial for ES-methods (Sections V-B and V-C).

II. Related Work

Research in robotic table tennis dates back to 1983 when Billingsley [4] proposed a robotic table tennis challenge with simplified rules and a smaller table. Subsequently, several early systems were developed [18, 11, 12]; see [28] for a summary of these approaches. At the time of the last competition in 1993, the problem remained far from solved.

Standard model-based approaches to robotic table tennis, as discussed in the introduction, consist of several steps: identifying virtual hitting points from trajectories, predicting ball velocities, calculating target paddle orientations and velocities, and finally generating robot trajectories leading to desired paddle targets. [25, 26, 2, 30, 47] take this approach and impose the additional constraint of fixing the intercept plane in the y-axis (perpendicular distance to the net). [13, 39, 21] allow for a variable ball intercept, but still use a virtual hitting point. The predictive ball model is either learned from data [25, 23, 24, 26] or is a parameterized dynamics model, which can vary in complexity from Newtonian dynamics with air drag [28, 30], to incorporating restitution and spin [47]. Robots vary from low-DOF robots with simple motion generation [25, 23, 24, 26] to anthropomorphic arms with strong velocity and acceleration constraints [28, 29, 42, 10].

Once a paddle target has been generated, the trajectory generation problem is still far from straightforward, especially if the robotic system has strong constraints. [28] resolve the redundancy in a 7DOF system by minimizing the distance to a 'comfort posture' in joint space whilst finding a paddle position and orientation that coincides with the hitting point. [29, 31] create a movement library of dynamical system motor primitives (DMPs) [15] from demonstrations, and learn to select from amongst them and generalize between them with their Mixture of Motor Primitives (MoMP) algorithm.

[14] and [19] take a different approach and do not identify a virtual hitting point. [14] use a combination of supervised and reinforcement learning to generate robot joint trajectories in response to ball trajectories. An important component of this system is a map which predicts the entire ball trajectory given the initial ball state estimated from a collection of measured ball positions. [19] use three ball models (flight model, ball-table rebound model, ball-racket contact) to generate a discrete set of ball positions and velocities, given ball observations queried from the vision system. Desired racket parameters for each set of ball positions are generated and then the optimization for robot joint movements is run. The system is fast enough to generate trajectories online.

[14] and [19] are the closest to our work. We use a similar anthropomorphic robotic system, a 6 DOF arm with revolute joints + 2 DOF linear axes, and do not make use of a virtual hitting point. However in our approach we do not use a predictive model of ball, nor do we use demonstrations to learn to generate trajectories.

To the best of our knowledge, two classes of model-free approaches have been applied to robotic table tennis. [47] frames the problem as a single-step bandit and uses DDPG [20] to predict the linear velocity and normal vector of the paddle given the position, linear and angular velocity of the ball at the hitting plane in simulation. [1] learns a local quadratic time-dependent Q-function from trajectory data, which they use to optimize a time-dependent stochastic linear feedback controller. The initial policy is learned from demonstration data. By contrast the main focus of this paper is on-policy methods, and our policies produce temporally extended actions at 100Hz in joint space.

III. Preliminaries

A. Robotic Table Tennis

Our goal in this paper is to train policies to solve the basic task of returning balls launched from the opponent side of the table. Beyond this, we are also interested in the *style* of the robot's play.

How a policy acts is as important as *how many* balls it can return, especially if the policy is used to control a physical robot. Good style – smooth control operating within the robot's limits – is crucial, in addition to the success rate.

Our policies should be able to execute complex playing styles involving high-level decision making, as humans do. In the context of robotic table tennis, an instance of this is *bimodal play*: the ability of the policy to select between *forehand* and *backhand* swings. Moreover, this should be *extensible*; in addition to the aforementioned goals of smoothness and bimodal play, the policy should have avenues for incorporating further styles or strategies as the difficulty of the task increases.

A key contribution of our paper is the development of methods to accomplish these goals using RL. We describe these methods in Section IV.

B. RL Background

To describe RL policies and algorithms, we use the formalism of the *Markov Decision Process* (MDP), which consists of a state space $\mathcal{S}$, action space $\mathcal{A}$, a reward function

Policy type	ES	PPO[1]
Layers	3	3
Channels (per layer)	8, 12, 8	8-32, 12-64, 8-32
Stride	1,1,1	1,1,1
Dilation	1,2,4	1,2,4
Activation function	tanh	tanh
Gating	Y,Y,N	Y,Y,Y
Total parameters	1.0k	2.4k - 36k

TABLE I
POLICY MODEL DETAILS.

$\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and stochastic transition dynamics $p(s_{t+1}|s_t, a_t)$. We parameterize a policy $\pi : \mathcal{S} \to \mathcal{A}$ as a neural network with parameters $\theta \in \Theta$, denoted as π_θ. The goal is to maximize the expected total reward $F(\theta) := \mathbb{E}_{a_t \sim \pi_\theta(s_t)} \left[\sum_{t=1}^{H} r(s_t, a_t)\right]$, where $a_t \sim \pi_\theta(s_t)$ indicates that actions follow π_θ.

Our controller π_θ generates continuous velocity commands in joint space. Within model-free RL, there are two broad classes of algorithms: those based on *value functions*, with the canonical example being Q-learning, and those using *direct policy search* [40]. Q-learning, which learns a function $Q(s, a)$ to predict the expected reward starting at state-action pair (s, a), has been successfully applied to manipulation problems [17]. Its disadvantage is that inference (i.e. selecting an action) involves solving an optimization problem (i.e. $\pi_\theta(s) = \text{argmax}_a Q(s, a)$) which can take several hundred milliseconds for continuous action spaces, as in [17] (using CMA-ES). This is impractical for high-speed tasks like table tennis. Instead, we opt for direct policy search methods, which learn a mapping from states to actions.

IV. Methods: Applying RL to Table Tennis

In this section, we describe our key design choices for successfully applying RL to robotic table tennis: the policy architecture, the use of curriculum learning, and the choice of optimization algorithm.

A. Policy Representation

We represent a policy using a three-layer convolutional neural network (CNN) with gated activation units [44] (Figure 2, Table I). We found the gating mechanism was important for accelerating training and varied the number of channels depending on the algorithm (ES, PPO) for best performance.

Figure 2 depicts this architecture. The input to the CNN is a tensor of shape $(T, S + 3)$, where T is the number of past observations, and S is the DOF of the robot. The $+3$ corresponds to a measurement of the 3D position of the ball, which is appended to the robot state. In our experiments, $S = 8$ and $T = 8$, so the input has shape $(8, 11)$. The CNN applies 1D convolutions on the time dimension, with the ball and joint state treated as channels. Each gated layer produces two tensors o_1 (red) and o_2 (yellow) of equal size. The gating mechanism then multiplies the activations $\tanh(o_1)$ elementwise with the mask $\sigma(o_2)$ to produce the output y_i.

[1]Architecture for both the value and policy networks. Total parameters are the sum of both networks' parameters.

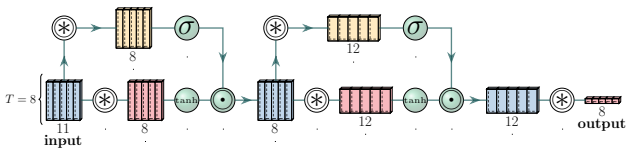

Fig. 2. Gated CNN architecture[3]. Rectangles represent channels, and 1D filters are applied on the vertical (time) dimension. Dilation reduces the height in successive layers. The symbol $\circledast$ denotes the application of convolutions, σ denotes elementwise sigmoid, and $\odot$ denotes elementwise (Hadamard) product. The dimensions shown are for the ES policy (Table I).

Formally, let W_i be the kernel of hidden layer i, b_i the bias, and X_i the input. We have

$$\begin{aligned}(o_1, o_2) &= X_i \circledast W_i + b_i \\ y_i &= \tanh(o_1) \odot \sigma(o_2).\end{aligned} \tag{1}$$

To better understand the effect of using a CNN controller, we also trained simple three-layer multi-layer perceptron (MLP) controllers with 50 and 10 units in two hidden layers.

B. Curriculum Learning in RL

The task of returning the ball defines a sparse reward of 1.0 given at the end of episodes (ball throws) in which the robot succeeded, and 0.0 otherwise. In principle, this type of sparse reward can be used to find an optimal policy as it exactly expresses the goal of the task. However, simply maximizing the success rate has two major drawbacks:

- Other considerations such as smoothness and style (see Section III-A) are not captured by the sparse reward.
- Sparse rewards make training difficult, as $\nabla F(\theta) = 0$ for a large portion of the parameter space. A large amount of exploration is required to observe any signal when rewards are highly sparse, which makes gradient estimation difficult. Furthermore, gradient-based algorithms are more likely to get trapped in *local maxima*.

We can address these issues using *curriculum learning* [3] by carefully adjusting two aspects of the problem:

- Shaping the *training distribution*, e.g. changing the distribution of ball throws during training so that a policy can improve on its weaknesses. In the MDP formalism for RL, this is changing the *distribution of initial states* $s_0 \sim \mathcal{P}(\mathcal{S})$.
- Shaping the *reward function*. New rewards can be added to e.g. discourage moving at high speeds, improve the swing pose, or to use forehand and backhand swings.

Curriculum learning is especially important for learning complex styles such as bimodal play, as we explore in Section V-F.

We briefly discuss an alternative to curriculum learning: human engineering of the desired behavior. Indeed, we can create a hierarchical policy which uses a fixed decision rule to select between sub-policies based on the predicted ball landing position. This is able to achieve a near-perfect success rate of 94% (Table VII, *Hierarchical*) by leveraging optimal forehand and backhand policies trained with RL.

[3]Drawn in PlotNeuralNet. https://github.com/HarisIqbal88/PlotNeuralNet

However this approach is ultimately limited. Though rules-based engineering is possible for achieving the narrow goal of selecting between forehand-backhand, it is highly likely that more complex aspects of style or strategy cannot be neatly decomposed into distinct modes. Moreover, training separate policies for each mode quickly increases the total number of parameters. This limits the viability of engineering (it is not *extensible*), and motivates our use of curriculum learning.

C. Algorithms

We consider two classes of RL optimization algorithms: *evolutionary search* and *policy gradient* methods.

1) Evolutionary Search (ES): This class of optimization algorithms [45, 33] has recently become popular for RL [36, 5, 6, 7]. ES is a general blackbox optimization paradigm, particularly efficient on objectives $F(\theta)$ which are possibly non-smooth. The key idea behind ES is to consider the *Gaussian smoothing* of F, given by the following equation:

$$F_\sigma(\theta) = \mathbb{E}_{\mathbf{g}\sim\mathcal{N}(0,\mathbf{I}_d)}[F(\theta+\sigma\mathbf{g})], \tag{2}$$

where $\sigma > 0$ controls the precision of the smoothing. It can be shown that $F_\sigma(\theta)$ is differentiable with gradients:

$$\nabla F_\sigma(\theta) = \frac{1}{\sigma}\mathbb{E}_{\mathbf{g}\sim\mathcal{N}(0,\mathbf{I}_d)}[F(\theta+\sigma\mathbf{g})\mathbf{g}], \tag{3}$$

for which it is easy to derive unbiased Monte Carlo (MC) estimators. The ES method applies stochastic gradient ascent (SGD) to $F_\sigma(\theta)$, using the estimator $\widehat{\nabla}F_\sigma(\theta)$, so the update takes the form $\theta \leftarrow \theta + \eta \cdot \widehat{\nabla}F_\sigma(\theta)$.

ES algorithms come in different variations, where the differentiation is made based on: Monte Carlo estimators used, additional performed normalizations, specific forms of control variate terms and more. We use ES methods with state normalization [32, 36], filtering [36], and reward normalization [22], with repeated rollouts for each parameter perturbation $\mathbf{g_i}$. We found that averaging the reward from repeated rollouts was important for training good policies. This is likely due to the high degree of random variation between episodes. We also observed that the state-normalization heuristic was crucial for achieving good performance.

2) Policy Gradient Methods: Policy gradient (PG) methods [46, 41, 40] are commonly used in RL and have been adapted for robotics [20]. We experiment with a state-of-the-art PG algorithm called *proximal policy optimization* (PPO) [37]. Note that standard PG methods require *stochastic* policies which is not the case for ES algorithms.

V. Experiments

Observations, results and conclusions from all conducted experiments are organized as follows:

- ES outperforms PPO in final reward, produces smoother policies, and requires fewer network parameters. (Section V-B).
- CNN-based architectures both outperform MLP-based architectures and produce significantly smoother motions (Section V-C).
- Reward shaping can improve the policy's smoothness without impacting the success rate (Section V-D).

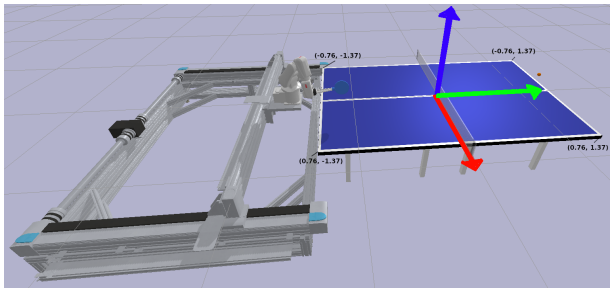

Fig. 3. Simulated robotic table tennis system. Our coordinate system places $(0, 0, 0)$ at the table center, and the axes are color-coded as x = red, y = green, z = blue.

- Bimodal play is nontrivial to learn, but can be obtained by using curriculum learning (Sections V-F and V-G). It does not require any modifications of the architecture.

A. System Description

Our robotic table tennis system consists of a 6DOF ABB IRB120 arm with revolute joints, and two linear axes permitting movement across and behind the table, for a total of 8DOF. The table conforms to ITTF standards and is 2.74m long, 1.525m wide, and the net is 15.25cm high. See Figure 3 for a depiction of the robot and table. The simulation is built using PyBullet [8].

Each episode consists of one ball throw, and the outcome is a *hit* if the paddle makes contact with the ball, and a *success* if it is hit and lands on the opponent's table. The simulation uses a simplified ball dynamics model that excludes air drag and spin. Ball throws are generated by randomly sampling the initial ball position[4] (x_0, y_0, z_0), the target landing coordinates on the robot side of the table $(x_1, y_1, 0)$, and the initial z-axis speed v_z. The full initial velocity vector (v_x, v_y, v_z) is then solved for, and we accept throws with $v_y \in [-8.5, -3.5]$. Using this approach, we define two ball distributions: a *forehand ball distribution* with $x_1 \in [-0.2, 0.7]$ and a *full table distribution* with $x_1 \in [-0.7, 0.7]$. At the start of each ball throw, the arm is initialized to either a forehand pose (see Figure 4, LHS) or a central pose (see: Figure 3) depending on the ball distribution. The initial pose is perturbed slightly to prevent overfitting.

For policy evaluation, we simulate 2500 episodes, and report success and hit rates, as well as smoothness metrics.

Randomness can be injected into the simulation in a number of different ways. Uniform random noise is added to the ball position at each timestep. Additionally, the ball and robot observations can be delayed independently and by a random number of simulation time steps each episode. Policy actions can also be delayed by a random number of time steps each episode.

B. ES and PPO

We first trained CNN-based ES and PPO policies on the forehand ball distribution with sparse rewards: +1 for hitting the ball, +1 for landing the ball. ES policies were each trained

[4]The coordinate system places $(0, 0, 0)$ at the table center, and the x and y axes are parallel to the width and length of the table. See Figure 3.

for 15K parameter updates, equivalently 22.7M episodes, where as the PPO policies were trained for 2M parameter updates, equivalently $1.1 - 1.2$M episodes.

ES policies were stronger in this setting, with 89% successful returns to the opponent side of the table (see Table II, Column S, "sparse" suffix). PPO policies attained only 70% success rate and required more parameters to train well on this task. For example, we found it was necessary to increase the number of parameters to $\sim$ 36K to achieve the best performance. PPO policies with a comparable number of parameters ($\sim$ 2.4K PPO vs $\sim$ 1.0K ES) only has a success rate of 22%. The phenomenon of policy gradient requiring more parameters than ES is not new, however. This effect has been observed numerous times in previous work. See for example: low-displacement rank policies with linear number of parameters or linear policies trained with ES outperforming PPO-trained ones [5, 22], as well as reverse trends between performance and architecture size for meta-learning (MAML) [38, 9]. Several explanations have been proposed previously for these effects, such as fewer parameters producing more stable and generalizable behaviors [22, 34]. Meanwhile, for policy gradient methods, complex architectural requirements such as requiring stronger representational power for value approximation [9], or additional batch normalization layers [20], can be bottlenecks.

Policy	S	H	J	A	V	JR
Random-8-12-8	0	2	1.5	1.0	0.9	6.1
ES-8-12-8-sparse	89	99	4.0	2.9	3.4	9.5
PPO-8-12-8-sparse	22	98	8.1	5.3	4.5	14.1
PPO-16-32-16-sparse	70	98	9.6	6.2	4.6	14.2
PPO-32-64-32-sparse	70	97	8.9	5.8	4.4	14.3
ES-8-12-8-shaped	**90**	99	2.9	2.2	2.9	9.2
ES-8-12-8-shaped + AF 3Hz	79	98	**2.0**	**1.4**	**2.1**	**9.0**
PPO-8-12-8-shaped	33	93	5.9	3.9	3.9	12.2
PPO-16-32-16-shaped	50	98	4.4	3.2	4.0	11.6
PPO-32-64-32-shaped	62	99	4.1	3.3	3.9	11.5

TABLE II
RESULTS: FOREHAND BALL DISTRIBUTION S: SUCCESS (%), H: HIT (%), J: AVG MAX JERK, A: AVG MAX ACCELERATION, V: AVG MAX VELOCITY, JR: SUM OF JOINT RANGE.[5]

We observe a similar performance differential between ES and PPO for the harder task of the full table ball distribution as shown in Table III. For this reason we conducted the ablation studies and training for bimodal play using only ES.

C. Policy architecture

Table IV shows that policy network architecture has a significant effect on both success rates and smoothness. We evaluate gated CNN and MLP policies, each with 3 hidden layers. Both policy types received the same inputs: the 8 most recent time-steps of ball and joint positions.

To evaluate smoothness we look at three metrics averaged over all time steps and joints: (1) maximum jerk per time step (J), (2) maximum acceleration per time step (A), and (3) maximum velocity per time step (V). We also measure the joint range (JR) of a policy.

[5]Bold indicates best performance in a column, excluding the random policy.

Policy	S	H	J	A	V	JR
Random-8-12-8	0	1	0.7	0.5	0.8	8.6
ES-8-12-8-sparse	39	97	5.5	3.6	3.6	10.9
PPO-8-12-8-sparse	18	89	6.3	4.5	3.7	11.1
PPO-16-32-16-sparse	21	94	7.0	4.9	4.4	14.5
PPO-32-64-32-sparse	34	95	9.2	6.0	4.4	14.4
ES-8-12-8-shaped	**48**	97	4.6	3.1	3.7	11.8
ES-8-12-8-shaped + AF 3Hz	8	98	2.2	2.0	2.7	11.8
PPO-8-12-8-shaped	1	52	**2.0**	**1.6**	**2.4**	**10.7**
PPO-16-32-16-shaped	4	**98**	4.1	3.3	4.0	14.4
PPO-32-64-32-shaped	31	96	4.2	3.9	4.2	13.6

TABLE III
RESULTS: FULL TABLE BALL DISTRIBUTION. S: SUCCESS (%), H: HIT (%), J: AVG MAX JERK, A: AVG MAX ACCELERATION, V: AVG MAX VELOCITY, JR: SUM OF JOINT RANGE.[5]

CNN policies achieved higher success rates in both the sparse reward and shaped reward (see V-D) settings compared to MLPs: 89% vs. 67% with sparse reward and 90% vs 46% with shaped rewards. MLP policies are significantly less smooth, with average max. jerk of 2.5 - 3x that of the CNN, 2.5x average max. acceleration, and 1.3 - 1.5x average max. velocity. MLPs also have a noticeably higher range of motion when compared with CNNs (column JR, Table IV). These differences are also clear from a qualitative inspection of the policy behavior (see policies B and C in the supplementary video).

Architecture	S	H	J	A	V	JR
CNN (sparse)	89	**99**	4.0	2.9	3.4	9.5
MLP (sparse)	67	98	11.1	7.2	4.4	11.4
CNN (shaped)	**90**	**99**	**2.9**	**2.2**	**2.9**	**9.2**
MLP (shaped)	46	95	7.6	5.4	4.3	13.4

TABLE IV
COMPARING CNN AND MLP POLICIES WITH SPARSE AND SHAPED REWARDS ON THE FOREHAND BALL DISTRIBUTION. ALL POLICIES WERE TRAINED FOR 15K PARAMETER UPDATES.

D. Reward shaping

Reward shaping is common practice in RL and it is an effective technique for shaping policy behavior. We explored the effect of seven different rewards (see footnote 6) shown in Table V, and find that it improves style with little or no cost in success rate for the same number of optimization steps.

Rewards[6]	S	H	J	A	V	JR
ST (sparse)	89	**99**	4.0	2.9	3.4	9.5
ST, IC, BBR	87	99	4.3	3.1	3.1	8.6
ST, IC, BBR, PH, JA	**91**	**99**	3.2	2.5	3.1	9.7
ST, IC, BBR, PH, V, A, J	96	**99**	**2.7**	**2.0**	**2.8**	**8.5**
Canonical (all rewards)[7]	90	**99**	2.9	2.2	2.9	9.2

TABLE V
ABLATION STUDY: EFFECT OF DIFFERENT REWARDS ON SUCCESS, HIT RATES, AND SMOOTHNESS METRICS. POLICIES WERE TRAINED AND EVALUATED ON THE FOREHAND BALL DISTRIBUTION.

[6]ST: hit and success sparse rewards, IC: penalty for self-collision or colliding with the table, BBR: penalty for rotating the base joint too far backwards, PH: penalty if the paddle gets too close to the table, JA: penalty if the arm position gets too close to any of the joint limits, V / A / JL: Penalty for exceeding a velocity / acceleration / jerk limit.

[7]Canonical reward shaping: ST, IC, BBR, PH, V, A, J, JA

We observe a similar pattern with PPO in that reward shaping improves the smoothness metrics (see Table II for example). However, PPO policies also are noticeably less smooth, with J, A, and V values over 2x that of comparable ES policies, and have a larger range of motion. It may be that PPO results in more sensitive and mobile policies compared to ES, and this makes it harder to train smooth policies. It would be interesting to explore this further in future work.

E. Action filters

We find that applying a low-pass butterworth filter to the policy actions further increases smoothness (see Table VI) but the differences in smoothness metrics between filters with varying cutoff frequencies is not very large. Unlike reward shaping, adding a filter appears to make the primary problem of returning balls harder, and leads to slightly lower success rates.

Action filter	S	H	J	A	V	JR
None	**90**	**99**	2.9	**2.1**	2.9	9.2
f-cut: 2Hz	84	97	**1.8**	**1.2**	2.2	9.0
f-cut: 3Hz	79	98	2.0	1.4	**2.1**	9.0
f-cut: 5Hz	80	**99**	1.9	1.3	2.2	**7.2**
f-cut: 10Hz	88	**99**	1.9	1.5	2.4	9.5

TABLE VI
ABLATION STUDY: EFFECT OF APPLYING A LOW-PASS ACTION FILTER TO THE POLICY OUTPUT. POLICIES WERE TRAINED AND EVALUATED ON THE FOREHAND BALL DISTRIBUTION.

F. Learning Complex Playstyles

Policy A, as shown in the video, is the result of training with a curriculum on the ball distribution and rewards. It has a success rate of 80% on the full table ball distribution, a hit rate of 99%, and has a good bimodal style, with balanced success rates on the forehand and backhand (Table VII). We also note that Policy A can obtain success rates nearing the human-engineered hierarchical policy (discussed in Section IV-B) which can be seen as a near-optimal policy for this problem, despite Policy A having a much smaller model.

Policy	S[8]	H	S-F	H-F	S-B	H-B
A	80	99	78	99	81	99
Hierarchical	94	100	92	100	96	100

TABLE VII
PERFORMANCE OF BIMODAL POLICIES ON THE FULL TABLE BALL DISTRIBUTION. POLICY A IS SHOWN IN VIDEO.

The curriculum used to train Policy A is listed in Section V-G. To understand the necessity of each component, we carry out a series of ablation studies, shown in Table VIII. In each ablation test, we train a policy for 15K steps and evaluate its success rate and style. Note that success rates are low overall, since training with the *full table distribution* (Section V-A) requires more steps than the *forehand table distribution* used in the ablation studies of Section V-B.

[8] 'S' denotes success rate, 'H' denotes hit rate, 'F' denotes the rate for ball throws on the forehand side of the table, and 'B' the backhand.

#	Regime	S[8]	H	S-F	H-F	S-B	H-B
	Pose Reward	(*see Section V-F1*)					
1	None	11.8	88.3	23.6	96.3	0.8	80.8
2	CPS	17.7	53.3	28.4	73.8	6.3	31.5
3	**DCPS**	2.3	83.7	1.4	85.4	3.3	82.1
4	**CPT**	0.8	95.3	1.3	95.2	0.4	95.5
	Success + Pose	(*see Section V-F2*)					
5	DTR + None	9.6	71.4	16.8	92.4	2.4	50.6
6	DTR + DCPS	11.3	88.3	18.9	98.0	2.7	77.3
	Ball Range + Pose	(*see Section V-F3*)					
7	$(0.5, 0.7)$ + None	16.4	36.9	32.6	73.6	0	0
8	$(0.3, 0.7)$ + None	0.3	38.3	0.6	77.0	0	0
9	$(0.3, 0.7)$ + CPS	3.6	19.1	7.0	37.5	0	0
10	$(0.3, 0.7)$ **+ DCPS**	8.5	87.6	9.7	87.4	7.3	87.5

TABLE VIII
ABLATION STUDIES: PERFORMANCE AFTER 15K STEPS FOR VARIOUS REGIMES. NUMBERS EXPRESS PERCENTAGES, AND BOLDED ROWS (**3,4, 10**) INDICATE BIMODAL POLICIES (FROM VISUAL INSPECTION).

A major challenge is that optimizing for success rates typically leads to a unimodal policy. For instance, in Table VIII, row #1 shows that under the canonical rewards, the resulting policy is forehand-only; it achieves 23% success on forehand balls but only 0.8% on backhand balls. This is also confirmed by visual inspection. To reliably obtain bimodal play, we must devise a curriculum to encourage it. First, we introduce *pose rewards* which can maintain bimodal style (Section V-F1). Then we discuss how adding success bonuses (Section V-F2) and shaping the task distribution can improve training (Section V-F3). Finally in Section V-G we present the curriculum for training Policy A.

1) Reward Shaping for Bimodal Play: We have already seen reward shaping in Section V-D to encourage better style. We use a similar technique for bimodal play.

Precisely characterizing 'forehand' swings and the ball throws for which a forehand swing should be rewarded, is nontrivial, and vary in the amount of human knowledge implicit in the reward definition. We consider several variants of pose rewards:

1) **Conditional Pose State** (CPS): We define reference poses for the forehand and backhand, which are shown in Figure 4. The reward is given for taking a pose close[9] to the reference pose corresponding to the side on which the ball lands. That is, $R^F_{CPS} = 1 - d^F$ is awarded for episodes where the ball lands on the forehand side, where d^F is the minimum distance of the arm's pose to the forehand reference point, and R^B_{CPS} is defined similarly for backhand.
2) **Dense Conditional Pose State** (DCPS): A denser version of the CPS reward which penalizes taking the wrong pose. The reward is $w(R^F_{CPS} - R^B_{CPS})$ for episodes where the ball is thrown to the forehand, and $w(R^B_{CPS} - R^F_{CPS})$ for throws to the backhand. We add a scale w which reduces the magnitude of the reward if the ball's landing point is near the center.
3) **Conditional Pose Timesteps** (CPT): We define forehand and backhand in terms of the rotation of J1 and J4 (Figure 1). The pose is considered to be 'forehand' if J1 and J4 both lie in the appropriate half of their ranges, and similarly for backhand. Let t^F to be the percentage

[9] As measured by the L_2 norm in joint space

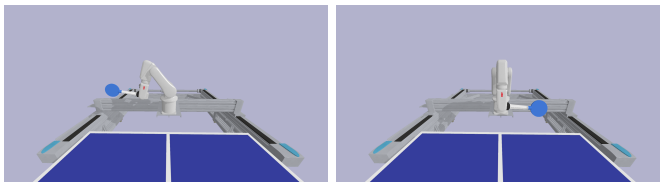

Fig. 4. Reference forehand (left) and backhand (right) poses.

of timesteps before ball contact such that the robot is in a forehand pose, and similarly for t^B on the backhand. The reward given is $R^F_{CPT} = w(t^F - t^B)$ for balls thrown to the forehand, and $R^B_{CPT} = w(t^B - t^F)$ for backhand balls.

We first investigate each type of pose reward (rows #1-4, Table VIII). As noted before, training without pose rewards led to a purely forehand policy (row #1). The CPS reward also led to a forehand policy (row #2), indicating that locally, the policy can still obtain higher reward by improving its forehand play at the expense of the backhand. This was a motivation for the DCPS reward (row #3). Although its success rate after the same period of training is lower, its hit rate is high, and the forehand and backhand success rates are balanced. Video inspection confirms it has a basic bimodal style. The CPT reward produces similar training as DCPS in early stages, but video inspection shows that DCPS produces a clearer bimodal style. In general, a more 'prescriptive' reward such as DCPS appears to speed up learning at the beginning (especially with distribution shaping; see Section V-F3), whereas a more 'flexible' reward such as CPT has advantages in later training (Section V-F2).

2) Reward Shaping to Escape Plateaus: Bimodal policies often reach a plateau in training where further progress becomes extremely slow. This is likely because of the sparsity of the success reward; since it does not distinguish between return balls that are close to landing and those that are far from the target, it is difficult for exploration to find nearby policies with sufficiently higher success rates so as to have measurably better rewards. We experiment with two rewards for increasing the success signal:

1) **Landing Bonus**: Increase the sparse success reward.
2) **Distance to Table** (DTR): A dense version of the success reward given by $\max\{1-d, -2\}$, where d is the minimum distance between the ball and the opponent's table surface during the return trajectory.

Introducing these rewards helps to escape plateaus for policies that are *already* bimodal and have reasonable success rates. Training such policies with the CPT pose reward and the DTR success reward leads to continued improvement while maintaining style, especially combined with distribution shaping (Section V-F3). However, we find that the balance between success rewards and pose rewards which lead to bimodal play can be sensitive. For example;

1) Using DTR at the onset leads to a unimodal policy, even with DCPS pose rewards (rows #5, 6).
2) When initialized with a bimodal policy (trained with DCPS), subsequent training with DTR and DCPS leads the policy to collapse to an unusual unimodal style which takes a backhand pose and then manipulates its 'shoulder' J3 (video, Policy E) to reduce penalties.
3) The combination of DCPS and a landing bonus of $+1.0$ leads to collapse.

3) Shaping the Training Distribution: Varying the distribution of tasks can improve training. For instance, if our objective is forehand play only, notice by comparing Table V to Table VIII that much higher success rates are obtained in 15K steps by restricting the ball distribution to forehand-only. Adjusting the difficulty of the task can therefore make training faster. It may also help to avoid local minima for which undesirable style is compensated by locally maximal success rates.

As we might expect, our experiments show that the policy has the greatest difficulty learning a bimodal style for balls that land at the center of the table. Based on this observation, we consider a spectrum of tasks where the ball's training distribution is supported on the two sides of the table. The *ball range* (a, b) for $0 \leq a < b$ indicates that the x-coordinate of the ball's landing position belongs to $[-b, -a] \cup [a, b]$. To overcome the style collapse problem when training a bimodal policy with the DTR reward, we change the ball range to $(0.5, 0.7)$ when the DTR reward is introduced, and then periodically widen the range as the policy improves until the distribution spans the entire table.

We also consider whether this distribution shaping can be applied at the onset. However, ablation studies indicate that the policy learns to ignore one side of the table entirely (see rows #7-9 of Table VIII), unless the DCPS reward is also given, which trains well. This shows that training is most efficient with a curriculum, which leads the policy to improve skills without forgetting previous ones.

G. Effective Curriculum Learning

Based on the observations of the preceding sections, we used the following scheme to train bimodal policies with RL, an example of which is Policy A.

1) Canonical style rewards and DCPS pose reward on the full table.
2) Introduce the DTR success reward, switch the pose reward to CPT, and set the ball range to $(0.5, 0.7)$.
3) Increase the ball range to $(0.3, 0.7)$, $(0.1, 0.7)$, and then to the full table.

The precise number of steps required varies owing to the randomness in RL algorithms; the first stage may take 15K steps, as was used in our ablation studies.

VI. Conclusions

We have shown that model-free reinforcement learning is an effective approach for robotic table tennis and is suitable for high-speed control, generating actions at 100Hz. This has the advantage of avoiding ball prediction modeling and trajectory optimization without the need for human demonstrations. Policies can learn to hit and return balls simply by being given sparse rewards for contact and success. Moreover, reward shaping and curriculum learning can

be used to improve the style of the policy, and develop more complex play. We demonstrated this by training strong bimodal policies which are capable of playing both the forehand and backhand. This suggests that RL is promising for robotic table tennis. Our experiments show that CNN policies outperform MLP policies both in terms of success and smoothness. We also observe that the evolution strategy (ES) paradigm in RL was particularly effective for this problem, and in comparison to policy gradient methods, was able to train smaller policies.

VII. Acknowledgements

We thank David D'Ambrosio, Jie Tan, and Peng Xu for their helpful comments on this work.

References

[1] Riad Akrour, Abbas Abdolmaleki, Hany Abdulsamad, Jan Peters, and Gerhard Neumann. Model-free trajectory-based policy optimization with monotonic improvement. *J. Mach. Learn. Res.*, 2016.

[2] Russell Anderson. *A Robot Ping-Pong Player: Experiments in Real-Time Intelligent Control*. MIT Press, 1988.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pages 41–48, 2009.

[4] John Billingsley. Robot ping pong. *Practical Computing*, 1983.

[5] Krzysztof Choromanski et al. Structured evolution with compact architectures for scalable policy optimization. In *ICML*, 2018.

[6] Krzysztof Choromanski et al. From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization. In *NeurIPS*, 2019.

[7] Krzysztof Choromanski et al. Provably robust blackbox optimization for reinforcement learning. In *CoRL*, 2019.

[8] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[9] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *ICLR*, 2018.

[10] Yapeng Gao, Jonas Tebbe, Julian Krismer, and Andreas Zell. Markerless racket pose detection and stroke classification based on stereo vision for table tennis robots. *IEEE Robotic Computing*, 2019.

[11] J. Hartley. Toshiba progress towards sensory control in real time. *The Industrial Robot 14-1*, pages 50–52, 1983.

[12] Hideaki Hashimoto, Fumio Ozaki, and Kuniji Osuka. Development of ping-pong robot system using 7 degree of freedom direct drive robots. In *Industrial Applications of Robotics and Machine Vision*, 1987.

[13] Yanlong Huang, Bernhard Schölkopf, and Jan Peters. Learning optimal striking points for a ping-pong playing robot. *IROS*, 2015.

[14] Yanlong Huang, Dieter Buchler, Okan KoÃğ, Bernhard Schölkopf, and Jan Peters. Jointly learning trajectory generation and hitting point prediction in robot table tennis. *IEEE-RAS Humanoids*, 2016.

[15] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. *ICRA*, 2002.

[16] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *CVPR*, 2019.

[17] Dmitry Kalashnikov et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRL*, 2018.

[18] John Knight and David Lowery. Pingpong-playing robot controlled by a microcomputer. *Microprocessors and Microsystems - Embedded Hardware Design*, 1986.

[19] Okan KoÃğ, Guilherme Maeda, and Jan Peters. Online optimal trajectory generation for robot table tennis. *Robotics & Autonomous Systems*, 2018.

[20] Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *ICLR*, 2016.

[21] Reza Mahjourian, Navdeep Jaitly, Nevena Lazic, Sergey Levine, and Risto Miikkulainen. Hierarchical policy design for sample-efficient learning of robot table tennis through self-play. *arXiv:1811.12927*, 2018.

[22] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *NeurIPS*, 2018.

[23] Michiya Matsushima, Takaaki Hashimoto, and Fumio Miyazaki. Learning to the robot table tennis task-ball control and rally with a human. *IEEE International Conference on Systems, Man and Cybernetics*, 2003.

[24] Michiya Matsushima, Takaaki Hashimoto, Masahiro Takeuchi, and Fumio Miyazaki. A learning approach to robotic table tennis. *IEEE Transactions on Robotics*, 2005.

[25] Fumio Miyazaki, Masahiro Takeuchi, Michiya Matsushima, Takamichi Kusano, and Takaaki Hashimoto. Realization of the table tennis task based on virtual targets. *ICRA*, 2002.

[26] Fumio Miyazaki et al. Learning to dynamically manipulate: A table tennis robot controls a ball and rallies with a human being. In *Advances in Robot Control*, 2006.

[27] Katharina Muelling and Jan Peters. A computational model of human table tennis for robot application. In *AMS*, 2009.

[28] Katharina Muelling, Jens Kober, and Jan Peters. A biomimetic approach to robot table tennis. *Adaptive Behavior*, 2010.

[29] Katharina Muelling, Jens Kober, and Jan Peters. Learning table tennis with a mixture of motor primitives. *IEEE-RAS Humanoids*, 2010.

[30] Katharina Muelling, Jens Kober, and Jan Peters. Simulating human table tennis with a biomimetic robot setup. In *Simulation of Adaptive Behavior*, 2010.

[31] Katharina Muelling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 2012.

[32] Anusha Nagabandi et al. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*, 2018.

[33] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *FoCM*, 2017.

[34] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham M. Kakade. Towards generalization and simplicity in continuous control. In *NeurIPS*, 2017.

[35] Marie-Marin Ramanantsoa and Alain Durey. Towards a stroke construction model. *The International Journal of Table Tennis Sciences*, 1994.

[36] Tim Salimans et al. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv:1703.03864*, 2017.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

[38] Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. ES-MAML: simple hessian-free meta learning. In *ICLR*, 2020.

[39] Yichao Sun, Rong Xiong, Qiuguo Zhu, Jingjing Wu, and Jian Chu. Balance motion generation for a humanoid robot playing table tennis. *IEEE-RAS Humanoids*, 2011.

[40] Richard Sutton and Andrew Barto. *Reinforcement Learning: An introduction*. MIT Press, MA, 1 edition, 1998.

[41] Richard Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*, 2000.

[42] Jonas Tebbe, Yapeng Gao, Marc Sastre-Rienietz, and Andreas Zell. A table tennis robot system using an industrial kuka robot arm. *GCPR*, 2018.

[43] Josh Tobin et al. Domain randomization and generative models for robotic grasping. In *IROS*, 2018.

[44] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *NeurIPS*, 2016.

[45] Daan Wierstra, Tom Schaul, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. In *2008 IEEE Congress on Evolutionary Computation*, 2008.

[46] Ronald Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

[47] Yifeng Zhu, Yongsheng Zhao, Lisen Jin, Jingjing Wu, and Rong Xiong. Towards high level skill learning: Learn to return table tennis ball using monte-carlo based policy gradient method. *IEEE International Conference on Real-time Computing and Robotics*, 2018.