i-Sim2Real: Reinforcement Learning of Robotic Policies in Tight Human-Robot Interaction Loops

Anonymous Author(s)
Affiliation
Address
email

Abstract: Sim-to-real transfer is a powerful paradigm for robotic reinforcement learning. The ability to train policies in simulation enables safe exploration and large-scale data collection quickly at low cost. However, prior works in sim-to-real transfer of robotic policies typically do not involve any human-robot interaction because accurately simulating human behavior is an open problem. In this work, our goal is to leverage the power of simulation to train robotic policies that are proficient at interacting with humans upon deployment. But there is a chicken and egg problem — how to gather examples of a human interacting with a physical robot so as to model human behavior in simulation without already having a robot that is able to interact with a human? Our proposed method, Iterative-Sim-to-Real (i-S2R), attempts to address this. i-S2R bootstraps from a simple model of human behavior and alternates between training in simulation and deploying in the real world. In each iteration, both the human behavior model and the policy are refined. We evaluate our method on a real world robotic table tennis setting, where the objective for the robot is to play *cooperatively* with a human player for as long as possible. Table tennis is a high-speed, dynamic task that requires the two players to react quickly to each other's moves, making for a challenging test bed for research on human-robot interaction. We present results on an industrial robotic arm that is able to cooperatively play table tennis with human players, achieving rallies of 22 successive hits on average and 150 at best. Further, for 80% of players, rally lengths are 70% to 175% longer compared to the sim-to-real (S2R) baseline.

Keywords: sim-to-real, human-robot interaction, reinforcement learning

1 Introduction

Sim-to-real transfer has emerged as a dominant paradigm for learning-based robotics. Real world training is often slow, cost-prohibitive, and poses safety-related challenges, so training in simulation is an attractive alternative and has been explored for a number of real world tasks, including object manipulation [1, 2, 3, 4], legged robot locomotion [5, 6], and aerial navigation [7, 8]. However, one element that is missing in this prior work is that the policies are not trained to be proficient at interacting with humans upon deployment. The utility of sim-to-real learning can be greatly increased if we extend it to settings where the trained policies need to interact with humans in a close, tight-loop fashion upon deployment. One of the major promises of learning-based robotics is to deploy robots in human-occupied settings, since non-learning robots already work well in deterministic, non-human occupied settings, such as factory floors. However, simulating human behavior is non-trivial (and indeed, one of the primary goals of artificial intelligence research), making it a major bottleneck in sim-to-real research for tasks involving human-robot interaction.

One approach to simulating human behavior is imitation learning: given a few examples of human behavior, we can use techniques such as behavior cloning [9, 10], or inverse reinforcement learning [11, 12] to distill that behavior into a policy, and then use these policies to generate human behavior in simulation. However, this approach presents a chicken and egg problem: in order to obtain useful examples of human behavior (in the context of human-robot interaction), we need a robot policy that already knows how to interact with humans in the real world, but we cannot

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

learn such a policy without the ability to simulate human behaviors in the first place. The primary contribution of this paper is a practical solution to this problem.

Our proposed method involves learning a coarse model of human behavior from initial data collected in the real world to bootstrap reinforcement learning of robotic policies in simulation. Deploying this learned policy in the real world now allows us to collect data in which the human subjects meaningfully interact with the robot. We then use this real world experience to improve our human behavior model, and continue training the robot policy in simulation under this updated model. We repeat this iterative process until a desired level of performance is achieved.

We present results on a task involving a robot playing table tennis with non-professional human players (see Figure 1). Table tennis is a high-speed, dynamic task that and requires close, tight-loop interactions between the two players (in this case, a human and a robot). We build an initial model of the human player's ball trajectories without a robot present and iteratively refine the robot and player models as they play together, ultimately resulting in a robot policy that can hold rallies of 22 successive hits on average and 150 at best.

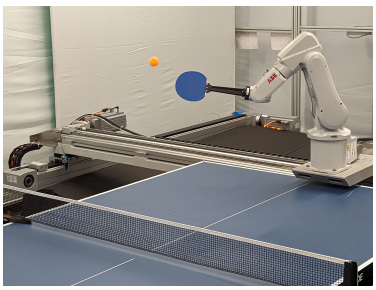

Figure 1: **Robot setup** An ABB IRB 120T 6-DOF robotic arm is mounted to a two-dimensional Festo linear actuator, creating an 8-DOF system.

While we demonstrate our approach on table tennis, we believe it is applicable more broadly, and can be applied to a number of tasks. For example, if the task involved a robot navigating through a busy hallway, we would first model the motion of human subjects alone (using motion capture devices, or a computer vision pipeline), and then train a policy in simulation with simulated human paths (so as to avoid collisions). Once this learned policy is deployed in the real world, the humans would likely alter their behavior in response to the robot, and capturing this data would allow us to create a more accurate human behavior model, which would further help us train a better policy. The process can be repeated until both human and robot behaviors converge, which would likely result in some co-adapted equilibrium point for the human and robot.

In summary, the primary contributions of this paper are: **(a)** a framework for training robotic policies in simulation that would need to interact with human subjects upon deployment, **(b)** a real world instantiation of this framework on a high-speed, dynamic task requiring tight, closed-loop interactions between humans and robots, **(c)** a detailed assessment of how our method, which we call Iterative-Sim-to-Real (i-S2R), compares with a baseline sim-to-real approach in the domain of cooperative robotic table tennis, and **(d)** the first robotic table tennis policy trained to control robot joints using reinforcement learning that can handle a wide variety of balls and can rally consistently with non-professional humans. To see videos of our system in action, please see the supplementary materials and this website `https://sites.google.com/view/i-s2r`.

2 Related Work

Sim-to-real Learning for Robotics Reinforcement Learning (RL) is a powerful paradigm for learning increasingly capable and robust robot controllers [13, 14, 15]. However, learning controllers from scratch on a physical robot is often prohibitively time consuming due to the large number of samples required to learn competent policies and potentially unsafe due to the random exploration inherent in RL methods [16, 17]. Training policies in simulation and transferring them to a physical robot, known as sim-to-real transfer (S2R), is therefore appealing.

Whilst it is both fast and safe to train agents from scratch in simulation, S2R presents its own challenge — persistent differences between simulated and real world environments that are extremely difficult to overcome [17, 18]. No single technique has been found to bridge the gap by itself. Instead a combination of multiple techniques are typically required for successful transfer. These include system identification [13, 19, 20, 21, 22] which may involve iterating with a physical robot in the loop [2, 23], building hybrid simulators with learned models [5, 13, 22], dynamics randomization [1, 2, 5, 6, 13, 14, 15], simulated latency [15, 22], and more complex network architectures [13]. We use (1) system ID with a physical robot in the loop, (2) dynamics randomization, (3) simulated latency, and (4) more complex networks. Similarly to Lee et al. [13], we use a 1D CNN to represent control policies. Yet a sim-to-real gap persists. Continuing to train in the real world [24, 25, 26] (known as

fine-tuning) is an effective way to bridge the remaining gap since the policy can adapt to changes in the environment. We also utilize fine-tuning in this work, but, unlike most past work, our learned policy is expected to interact cooperatively with a real human during this fine-tuning phase.

The closest sim-to-real approaches in prior work are Chebotar et al. [2] and Farchy et al. [23] since they update simulation parameters based on multiple iterations of real world data collection interleaved with simulated training. However, both of these prior works focus on using real world interaction data to learn improved *physical* parameters for the simulator, whereas our method focuses on learning better human behavior models. Unlike these prior work, our learned policies are proficient at interacting with humans upon deployment in the real world.

Reinforcement Learning for Table Tennis Robotic table tennis is a challenging, dynamic task [27] that has been a test bed for robotics research since the 1980s [28, 29, 30, 31, 32]. The current exemplar is the Omron robot [33]. Until recently, most methods tackled the problem by identifying a virtual hitting point for the racket [34, 35, 36, 37, 38, 39, 40, 41]. These methods depend on being able to predict the ball state at time t either from a ball dynamics model which may be parameterized [34, 35, 42, 43] or by learning to predict it [32, 37, 38]. This results in a target paddle state or states and various methods are used to generate robot joint trajectories given these targets [32, 34, 35, 42, 43, 44, 45, 46, 47, 48, 49]. More recently, Tebbe et al. [50] learned to predict the paddle target using RL.

An alternative line of research seeks to do away with hitting points and ball prediction models, instead focusing on high frequency control of a robot's joints using either RL [27, 38, 51] or learning from demonstrations [45, 52, 53]. Of these, Büchler et al. [27] is the most similar, training RL policies to control robot joints from scratch at high frequencies given ball and robot states as policy inputs. However Büchler et al. [27] restricts the task to playing with a ball thrower on a single setting, whereas we focus on the harder problem of cooperative play with different humans.

Most prior work simplifies the problem by focusing on play with a ball thrower. Only a few [45, 48, 50, 54] focus on cooperative rallying with a human. Of these, Tebbe et al. [50], is the most similar, evaluating policies on various styles of human-robot cooperative play. However, Tebbe et al. [50] simplify the environment to a single-step bandit and the policy learns to predict the paddle state given the ball state at a pre-determined hit time t. In contrast, we learn closed-loop policies that operate at a high frequency (75Hz), removing the need for learned policy to accurately predict where the ball will be in the future, increasing the robustness of the system, and enabling more dynamic play.

Human Robot Interaction Although not a typical HRI benchmark, cooperative robotic table tennis exhibits many of the features studied in the field: a human and robot working together, complex interactions between the two, inferring actions based on non-explicit cues, and so on. A major challenge in HRI is effectively modeling the complexities of human behavior in simulation [55] in order to learn without requiring an actual human. We employ several common techniques from HRI to learn in simulation such as simplifying the human model [56], specialized models for specific players [57], and refining our model based on real world interactions. Finally we note that like us, Paleja et al. [58] found policy performance varied depending on the skill of the human player.

3 Preliminaries

Problem Setting We consider the problem of a cooperative human-robot table tennis as a single-agent sequential decision making problem in which the human is a part of the environment. We formalize the problem as a *Markov Decision Process* (MDP) [59] consisting of a of a 4-tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$, p), whose elements are the state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and transition dynamics $p : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. An episode $(s_0, a_0, r_0, ..., s_n, a_n, r_n)$ is a finite sequence of $s \in \mathcal{S}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$ elements, beginning with a start state s_0 and ending when the environment terminates. We define a parameterized policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$ with parameters θ. The objective is to maximize $\mathbb{E}\left[\sum_{t=1}^{N} r(s_t, \pi_\theta(s_t))\right]$, the expected cumulative reward obtained in an episode under π_θ.

We make two simplifications to our problem. First, we focus on rallies starting with a hit instead of a table tennis serve to make the data more uniform. Second, an episode consists of a single ball throw and return. Policies are therefore rewarded based on their ability to return balls to the opposite side of the table. This reward structure encourages longer rally length, as an agent that can return any ball can also rally indefinitely provided the simulated single shots overlap with the real rally shots.

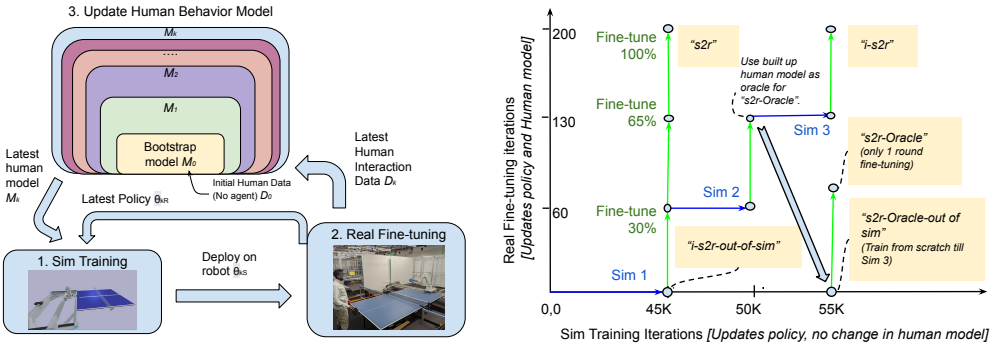

Figure 2: **Iterative-Sim-to-Real. left** We start with a coarse bootstrap model of human behavior (shown in yellow), and use it to train an initial robot policy in simulation. We then fine-tune this policy in the real world against a human player, and the human interaction data collected during this period is used to update the human behavior model used in simulation. We then take the fine-tuned policy back to simulation to further train it against the improved human behavior model, and this process is repeated until robot and human behaviors converge. **right** Specific i-S2R details used in this work. x-axis represents the training iterations in sim, y-axis represents the fine-tuning iterations in real with human-in-the-loop. Model names are in *italics*

Evolutionary Strategies Our proposed approach can be used with any RL algorithm, but we optimize our policies using evolutionary strategies (ES) [60, 61, 62, 63, 64] which have been shown to be an effective strategy for solving MDPs [62, 64]. The main idea behind ES is to maximise the Gaussian smoothing of the RL objective described above. Let $F(\theta)$ be the RL objective where θ are the policy parameters, then the ES objective is given by:

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

where $\sigma > 0$ controls the precision of the smoothing, and δ is a random normal vector with the same dimension as the policy parameters θ. We apply common ES approaches such as state normalization [62, 65], reward normalization [64], and perturbation filtering [62]. We also repeat and average rollouts with the same parameters to reduce variance. See Appendix A for details.

4 Method

i-S2R consists of two core components: **(1)** an iterative procedure for progressively updating and learning from a human behavior model — the human ball distribution in this setting — and **(2)** a method for modeling human behavior in simulation given a dataset of human play gathered in the real world (see Figure 2 for an overview). We first describe our iterative training procedure, and then discuss how we model human ball distributions.

Iterative training procedure An overview of the method can be seen in Figure 2. First we gather an initial dataset, D_0, from player P hitting table tennis balls across the table without a robot present. From D_0, we build our first human behavior model M_0 that defines a ball distribution (see below). A robot policy is trained in simulation to return balls sampled from M_0. Once the policy has converged, we transfer the parameters, θ_{0S}, to a real robotic system. The model is fine-tuned whilst player P plays cooperatively (i.e. trying to maximize rally length) with the robot for a fixed number of parameter updates to produce θ_{0R}. All of the human hits during this fine-tuning phase are added to D_0 to form D_1, which is used to define M_1. The policy weights, θ_{0R}, are then transferred back to simulation and training is continued with the new distribution M_1. After training in sim, the policy weights θ_{1S} are transferred back to the real world. The fine-tuning process is repeated to produce the next set of policy parameters θ_{1R}, dataset D_2, and human model M_2. This process can be repeated as many times as needed. One useful method for knowing when to stop is to check the change in human model in each iteration. See Appendix B for more details.

Modeling human ball distributions One of our primary goals is to simulate human player behaviors from a set of real world ball trajectories that have been subjected to air drag, gravity, and spin. Due to perception challenges in the real world, we do not explicitly model spin. The input to this procedure is a dataset of ball trajectories, where each trajectory consists of a sequence of ball

positions. The output is a uniform ball distribution defined by 16 numbers: the min and max initial ball position (6), velocity (6), and x and y ball landing locations on robot side (4).

The ball distribution is derived from the dataset in two stages. The first step is to estimate a ball's initial position and velocity for each trajectory. We do this by selecting the free flight part of the trajectory (before the first bounce) and minimize the Euclidean distance between the simulated and real trajectory using the Nelder-Mead method [66]. We use the model, $\ddot{x}_t = g - K_d||\dot{x}_t||\dot{x}_t$, $x_{t+1} = x_t + \Delta t(\dot{x}_t + \frac{\Delta t \ddot{x}_t}{2})$, $\dot{x}_{t+1} = \dot{x}_t + \Delta t \ddot{x}_t$ to simulate a trajectory, where (1) x_t, $\dot{x}_t$, and $\ddot{x}_t$ denote the position, velocity, and acceleration of the ball at time t, (2) $g = -9.81m/s^2[0, 0, 1]^T$ is the gravity, and (3) $K_d = C_d \rho \frac{A}{2m}$. $m = 0.0027kg$ is the ball's mass, $\rho = 1.29kg/m^3$ is the air density, $C_d = 0.47$ is the the drag coefficient, and $A = 1.256 \times 10^{-3} m^2$ is the cross-sectional area for a standard table tennis ball.

We remove outliers using DBSCAN [67] and take the minimum and maximum per dimension to define the ball distribution. We sample an initial position and velocity from this distribution and generate a ball trajectory in simulation subject to the drag force. Other parameters needed for the simulation, such as coefficient of restitution, friction between the table and ball and the robot paddle and the ball, and so on have been empirically estimated following [68, 69].

5 System, Simulation, and MDP Details

Our real world robotic system (see Figure 1) is a combination of an ABB IRB 120T 6-DOF robotic arm mounted to a two-dimensional Festo linear actuator, creating an 8-DOF system, with a table tennis paddle mounted on the end-effector. The 3D ball position is estimated via a stereo pair of Ximea MQ013CG-ON cameras from which we process 2D detections, triangulate to 3D, and filter through a 3D tracker. See Appendix C for more details. We concatenate the ball position with the 8-DOF robot joint angles to form an 11-dimensional observation space. Along with the current observation, we pass the past seven observations (a state space of 8×11) as an input to the policy. The policy controls the robot by outputting eight individual joint velocities at 75Hz. Following Gao et al. [51] we use a 3-layer 1-D dilated gated convolutional neural network as our policy architecture. Details of the policy architecture can be found in Appendix D.

Our simulation is built on the PyBullet [70] physics engine replicating our real environment. We use PyBullet to model robot and contact dynamics whilst balls are modeled as described in Section 4. We add random uniform noise of $2\times$ the diameter of a table tennis ball to the ball observation per timestep to aid transfer to a physical system. We also found it necessary to simulate sensor latency, otherwise sim-to-real transfer completely failed. Robot actions as well as ball and robot observation latencies are modeled as parameterized Gaussians based on measurements from the real system. Policies are rewarded for hitting balls and for returning balls in a cooperative manner. See Appendix F for more details.

6 Experimental Results

Here we aim to answer the following questions; **(1)** Does i-S2R improve over baseline sim-to-real with fine-tuning (which we refer to as S2R) in a human-robot interactive setting where the human behavior changes in response to the robot policy? **(2)** How many sim-to-real iterations does the human behavior model need to converge? **(3)** How much of i-S2R's performance can be attributed to (a) improving the human behavior model vs. (b) the additional training steps in simulation? and **(4)** Does i-S2R generalize better to new players compared with S2R?

Experimental setup To evaluate our method, we completed the procedure described in Section 4 for five different non-professional table tennis players, thus training five independent i-S2R policies. Each player also trained (1) a S2R baseline which was given the same amount of real world training time as the i-S2R policy and (2) a S2R-Oracle which was trained in simulation on the penultimate human behavior model obtained through i-S2R and fine-tuned in the real world for 35% of the i-S2R training budget. This is equivalent to the last round of fine-tuning for i-S2R. (See Figure 2 **right**). S2R-Oracle is intended to isolate the effect of the human behavior modeling on final performance, enabling us to better understand what aspects of the i-S2R process matter.

Each model was evaluated by (a) the model's trainer and (b) two other players. In each evaluation, 50 rallies (defined as a sequence of consecutive hits ending when one player fails to return the ball)

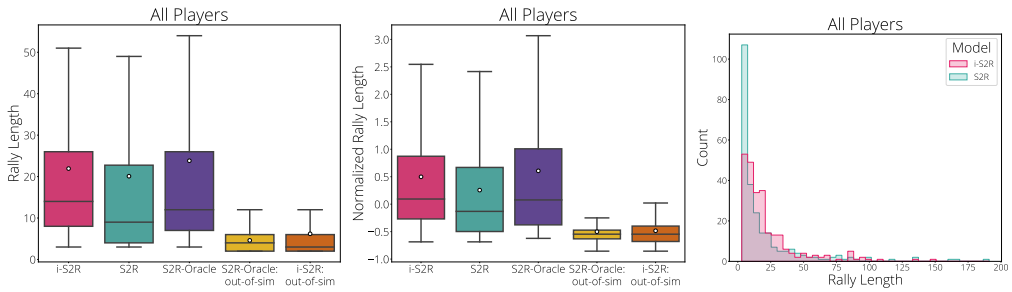

Figure 3: **Aggregated results** *Boxplot details:* The white circle is the mean, the horizontal line is the median, box bounds are the 25th and 75th percentiles. "out-of-sim" refers to models that are deployed on the real hardware with zero fine-tuning (see Figure 2). **left** When aggregated across all players, i-S2R rally length is higher than S2R by about 9%. However, note that simple aggregation puts extra weight on higher skilled players that are able to hold a longer rally. **center** The normalized rally length distribution (see Appendix I for normalization details) shows a bigger improvement between i-S2R and S2R in terms of the mean, median and 25th and 75th percentiles. **right** The histogram of rally lengths for i-S2R and S2R (250 rallies per model) shows that a large fraction of the rallies for S2R are shorter (i.e. less than 5), while i-S2R achieves longer rallies more frequently.

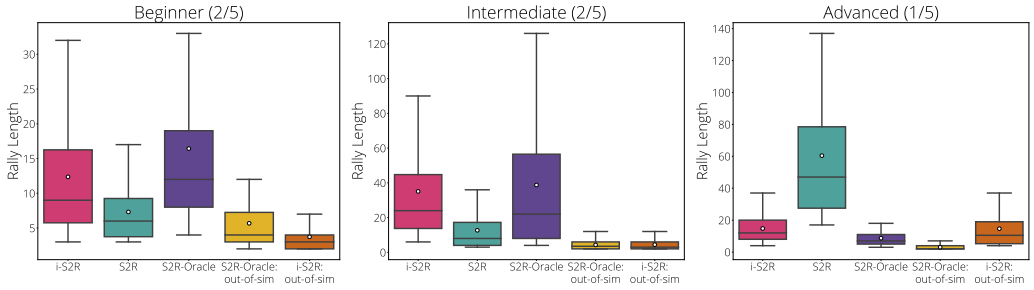

Figure 4: **Results by player skill.** When broken down by player skill, we notice that i-S2R has a significantly longer rally length than S2R and is comparable to S2R-Oracle for beginner and intermediate players. The advanced player is an exception. Note that S2R-Oracle gets just 35% of i-S2R and S2R fine-tuning budget.

were played with the human always starting and the rally length calculated as the number of paddle touches for both the human and robot. While the human can be responsible for a rally ending, almost all ended with the robot failing to return the ball or returning it such that the human could not easily continue the rally. The model trainer also evaluated intermediate checkpoints (see Figure 2) using the same methodology to shed light on the training dynamics. To ensure fair evaluation, all models were tested in random order and the identity of the model was kept hidden from the evaluator (*"blind eval"*). Further details can be found in Appendix G.

Due to the time needed to train and evaluate i-S2R, S2R, and S2R-Oracle (roughly 20 hours per person) we note that 4 of the 5 players are authors on this paper. The non-author player's results appear consistent with our overall findings (see Appendix J for details).

(1) Does i-S2R improve over S2R in a human-robot interactive setting? Figure 3 presents rally length distributions aggregated across all players whilst Figure 4 splits the data by skill. Players are grouped in to beginner (40% players), intermediate (40% of players) and advanced (20% players). The non-author player was classified as beginner. Please see Appendix H for skill level definitions. When aggregated over all players, we see that i-S2R is able to hold longer rallies (i.e. rallies that are longer than length 5) at a much higher rate than S2R, as shown in Figure 3. When the players are split by skill level, i-S2R significantly outperforms S2R for both beginner and intermediate players (80% of the players). The improvement differs between the two groups, with i-S2R yielding a $\approx 70\%$ and $\approx 175\%$ improvement for beginner and intermediate players respectively.

The policy trained by the advanced player has a different trend. Here, S2R dramatically outperforms i-S2R. We hypothesize that a good out-of-sim model (after 1st round of sim training) plays a large part in this difference (see Figure 5). However, due to the time consuming nature of repeating experiments on the physical system it is difficult to fully explain why this is the case, especially since both the training methodology and involvement of humans introduces a high degree of variance.

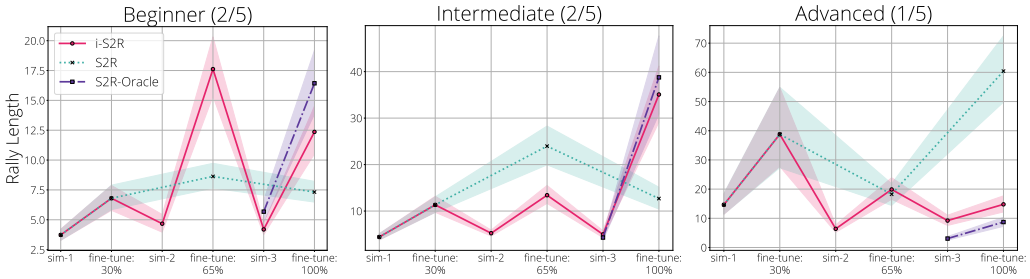

Figure 5: Policy performance at key checkpoints during training. For beginner players i-S2R performance converges after just two iterations (see fine-tune-65%). For intermediate players i-S2R takes three iterations to converge (see fine-tune-100%). "S2R-Oracle-sim-3" here is same as "S2R-Oracle-out-of-sim" in Figure 4.

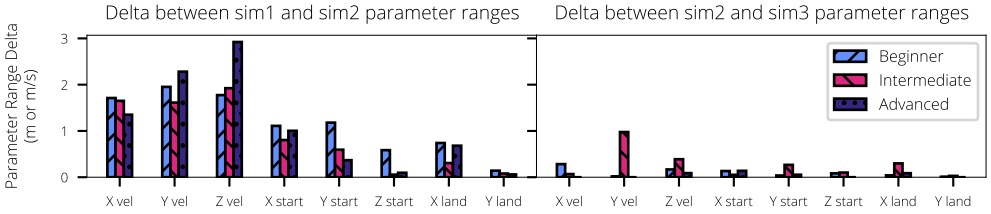

Figure 6: While the key distribution parameters change significantly from initial ball distribution (sim1) to that after 1st round of sim training (sim2), the change in the parameters between 1st and 2nd round of sim training is much less (sim2 vs. sim3).

(2) How many sim-to-real iterations does the human behavior model take to converge? For beginners we find that it only took two iterations for i-S2R to converge (see Figure 5). In the leftmost chart showing beginner policy data, i-S2R achieves comparable levels of performance at the end of the 2nd (fine-tune-65%) and final (fine-tune-100%) iterations. However, for intermediate skilled players this is not the case. The change in the human behavior model (ball distribution) from iteration to iteration shown in Figure 6 offers a clue. For beginner players, the distribution barely changes after the 2nd round as evidenced by the difference between the left and right charts. Whereas for intermediate players the distribution continues to change substantially from round 2 to 3 (specifically in y and z velocities), which is perhaps why we see the strongest performance of i-S2R after the 2nd iteration for beginners but after the 3rd iteration for intermediate players.

The advanced player's distribution hardly changes between the 2nd and 3rd round and the performance of i-S2R is comparable across both. However this does not explain why we observed the best i-S2R performance at the end of the 1st round for this player. We hypothesize that a good out-of-sim model after first round of training (see Figure 5) plays a large part in this. Investigating the effect of playing style on changes in ball distribution every iteration and hence on the sim-to-real gap or training for more iterations for advanced players can shed light on this in future work.

(3) What is the impact of the human behavior model? For beginner and intermediate players, S2R-Oracle is in line with i-S2R performance. However S2R-Oracle also achieved this level of performance with just 35% of the real world training time compared to i-S2R and S2R. Therefore much of the benefit of i-S2R likely comes from improving the human behavior model from iteration to iteration. It also suggests that if we had access to the final human behavior model at the beginning of training, the iterative sim-to-real training would not be needed. We could simply fine-tune in real and achieve comparable performance with significantly less human training time. S2R-Oracle's strong performance also validates our motivation for this work, in which we hypothesized that the difficulty of defining a good human behavior model a priori for human-robot cooperative rallies was limiting performance.

This result suggests that i-S2R does not benefit from additional training iterations in simulation over and above the improvements to the human behavior model. The evaluations at earlier stages in training (shown in Figure 5) suggest the remaining sim-to-real gap could be responsible. Figure 5 shows that, in all cases, after both the second (sim-2) and third (sim-3) rounds of simulated training, rally length drops noticeably. Reducing the sim-to-real gap might improve i-S2R's performance due to better starting points for the last two rounds of fine-tuning.

(4) Does i-S2R offer any generalization benefits in this setting? We now evaluate the generalization capabilities of models trained with i-S2R, and how they compare against models trained using S2R. As shown in Figure 7, i-S2R significantly outperforms S2R when the models are cross evaluated by other players (with similar blind evaluations as earlier) including for the advanced player where S2R was best in self evaluation (see Appendix J for details by player). This observation holds whether we look at absolute or normalized rally length (see Appendix I normalization methodology details). Performance with other players is lower for all models, however i-S2R maintains around 70% of performance on average compared to 30% for S2R. We hypothesize that the broader training distribution obtained by iterating between simulation and reality leads to policies that can deal with a wider range of ball throws, leading to better generalization to new players. Our confidence in this hypothesis is strengthened by the fact that both i-S2R and S2R-Oracle significantly outperform S2R under this setting.

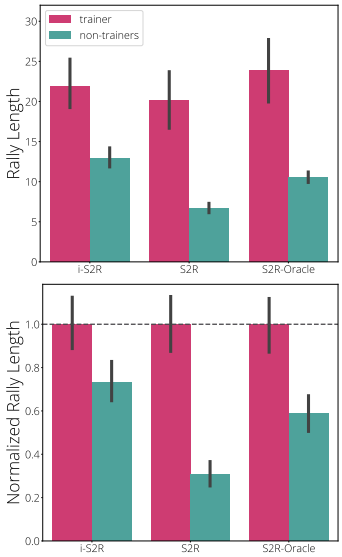

Figure 7: Cross-evaluations mean rally lengths (with 95% CI) aggregated across all players. i-S2R generalizes better to new players compared to S2R.

7 Limitations

Having a human in the loop poses numerous challenges to robotic reinforcement learning. It slows down the overall learning process to accommodate human participants, and limits the scale at which one can experiment. As one example, while we tested our method on five subjects, time limitations prevented us from training with multiple random seeds for each subject. There is significant variation in how people interact with robots (or sometimes even the same person over time), which introduces extra variance into our experiments. In our experiments, the trends we saw for one particular subject were substantially different from all other subjects, and we could not come to a clear explanation of why that was the case.

It is possible for an expert human player to get long rallies by keeping the ball in a very narrow distribution without really improving the inherent capability of the agent to play beyond those balls. In our studies, since we used non-professional players, this was not an issue. However, for future work in cooperative human-robot tasks, it would be interesting to explore ways to disentangle the skill level of the robot from the human participant.

Another limitation arising from training a policy with a human in the loop is the possibility that some performance improvements are attributable to human learning and not policy learning. We did our best to mitigate this by asking players to evaluate all models "blind" (i.e. the player is unaware of what model they are evaluating) and at the end of training, after which the majority of human learning was likely to have occurred. Consequently, we think that differences between models reflect differences in policy capability and not human.

Finally, we represent humans in simulation in a simple way — by capturing all initial position and velocity ranges during their play — and then we sample each ball in simulation uniformly and independently. This ignores the probability distribution of balls within those ranges and also results in a loss of correlation between subsequent balls in a rally. This could be addressed by developing a more sophisticated ball model that takes these factors into account.

8 Conclusion

We present i-S2R to learn RL policies that are able to interact with humans by iteratively training in simulation and fine-tuning in the real world with humans in the loop. The approach starts with a coarse model of human behavior and refines it over a series of fine-tuning iterations. The effectiveness of this method is demonstrated in the context of a table tennis rallying task. Extensive "blind" experiments shed light on various aspects of the method and compare it against a baseline where we train and fine-tune in real only once (S2R). We show that i-S2R outperforms S2R in aggregate, and the difference in performance is particularly significant for beginner and intermediate players (4/5). Moreover, i-S2R generalizes much better than S2R to other players.

References

[1] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 1–8. IEEE, 2018.

[2] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. D. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 8973–8979. IEEE, 2019.

[3] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *Int. J. Robotics Res.*, 39(1), 2020.

[4] S. Kataoka, S. K. S. Ghasemipour, D. Freeman, and I. Mordatch. Bi-manual manipulation and attachment via sim-to-real reinforcement learning, 2022. URL https://arxiv.org/abs/2203.08277.

[5] J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Sci. Robotics*, 4(26), 2019.

[6] X. B. Peng, E. Coumans, T. Zhang, T. E. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. In M. Toussaint, A. Bicchi, and T. Hermans, editors, *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*, 2020.

[7] F. Sadeghi and S. Levine. CAD2RL: real single-image flight without a single real image. In N. M. Amato, S. S. Srinivasa, N. Ayanian, and S. Kuindersma, editors, *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*, 2017.

[8] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Learning high-speed flight in the wild. *Sci. Robotics*, 6(59), 2021.

[9] D. Pomerleau. ALVINN: an autonomous land vehicle in a neural network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1, [NIPS Conference, Denver, Colorado, USA, 1988]*, pages 305–313.

[10] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *arXiv preprint arXiv:1710.04615*, 2017.

[11] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In C. E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

[12] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In D. Fox and C. P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1433–1438. AAAI Press, 2008.

[13] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *CoRR*, 2020.

[14] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand. 2019.

[15] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *CoRR*, abs/1804.10332, 2018. URL http://arxiv.org/abs/1804.10332.

[16] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. *CoRR*, 2020.

[17] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. Karen Liu, J. Peters, S. Song, P. Welinder, and M. White. Sim2real in robotics and automation: Applications and challenges. *IEEE Transactions on Automation Science and Engineering*, 18(2):398–400, 2021. doi:10.1109/TASE.2021.3064065.

[18] M. Neunert, T. Boaventura, and J. Buchli. Why off-the-shelf physics simulators fail in evaluating feedback controller performance - a case study for quadrupedal robots. 2016.

[19] S. Zhu, A. Kimmel, K. E. Bekris, and A. Boularias. Model identification via physics engines for improved policy search. *CoRR*, 2017.

[20] M. Kaspar, J. D. M. Osorio, and J. Bock. Sim2real transfer for reinforcement learning without dynamics randomization. *CoRR*, abs/2002.11635, 2020.

[21] J. Tan, Z. Xie, B. Boots, and C. K. Liu. Simulation-based design of dynamic controllers for humanoid balancing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2729–2736, 2016. doi:10.1109/IROS.2016.7759424.

[22] K. Ota, D. K. Jha, D. Romeres, J. van Baar, K. A. Smith, T. Semitsu, T. Oiki, A. Sullivan, D. Nikovski, and J. B. Tenenbaum. Towards human-level learning of complex physical puzzles. *CoRR*, abs/2011.07193, 2020. URL https://arxiv.org/abs/2011.07193.

[23] A. Farchy, S. Barrett, P. MacAlpine, and P. Stone. Humanoid robots learning to walk faster: From the real world to simulation and back. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '13, page 39–46, 2013.

[24] S. Barrett, M. E. Taylor, and P. Stone. Transfer learning for reinforcement learning on a physical robot. In *AAMAS 2010*, 2010.

[25] A. A. Rusu, M. Vecerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. *CoRR*, 2016.

[26] X. Song, Y. Yang, K. Choromanski, K. Caluwaerts, W. Gao, C. Finn, and J. Tan. Rapidly adaptable legged robots via evolutionary meta-learning. *CoRR*, 2020. URL https://arxiv.org/abs/2003.01239.

[27] D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters. Learning to play table tennis from scratch using muscular robots. *CoRR*, abs/2006.05935, 2020. URL https://arxiv.org/abs/2006.05935.

[28] J. Billingsley. Robot ping pong. *Practical Computing*, 1983.

[29] J. Knight and D. Lowery. Pingpong-playing robot controlled by a microcomputer. *Microprocessors and Microsystems - Embedded Hardware Design*, 1986.

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

[31] H. Hashimoto, F. Ozaki, and K. Osuka. Development of ping-pong robot system using 7 degree of freedom direct drive robots. In *Industrial Applications of Robotics and Machine Vision*, 1987.

[32] K. Muelling, J. Kober, and J. Peters. A biomimetic approach to robot table tennis. *Adaptive Behavior*, 2010.

[33] A. Kyohei, N. Masamune, and Y. Satoshi. The ping pong robot to return a ball precisely. 2020.

[34] F. Miyazaki, M. Takeuchi, M. Matsushima, T. Kusano, and T. Hashimoto. Realization of the table tennis task based on virtual targets. *ICRA*, 2002.

[35] F. 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.

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

[37] K. Muelling et al. Simulating human table tennis with a biomimetic robot setup. In *Simulation of Adaptive Behavior*, 2010.

[38] Y. Zhu, Y. Zhao, L. Jin, J. Wu, and R. 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.

[39] Y. Huang, B. Schölkopf, and J. Peters. Learning optimal striking points for a ping-pong playing robot. *IROS*, 2015.

[40] Y. Sun, R. Xiong, Q. Zhu, J. Wu, and J. Chu. Balance motion generation for a humanoid robot playing table tennis. *IEEE-RAS Humanoids*, 2011.

[41] R. Mahjourian, N. Jaitly, N. Lazic, S. Levine, and R. Miikkulainen. Hierarchical policy design for sample-efficient learning of robot table tennis through self-play. *arXiv:1811.12927*, 2018.

[42] M. Matsushima, T. Hashimoto, and F. Miyazaki. Learning to the robot table tennis task-ball control and rally with a human. *IEEE International Conference on Systems, Man and Cybernetics*, 2003.

[43] M. Matsushima, T. Hashimoto, M. Takeuchi, and F. Miyazaki. A learning approach to robotic table tennis. *IEEE Transactions on Robotics*, 2005.

[44] K. Muelling, J. Kober, and J. Peters. Learning table tennis with a mixture of motor primitives. *IEEE-RAS Humanoids*, 2010.

[45] K. Muelling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 2012.

[46] Y. Huang, D. Buchler, O. Koç, B. Schölkopf, and J. Peters. Jointly learning trajectory generation and hitting point prediction in robot table tennis. *IEEE-RAS Humanoids*, 2016.

[47] O. Koç, G. Maeda, and J. Peters. Online optimal trajectory generation for robot table tennis. *Robotics & Autonomous Systems*, 2018.

[48] J. Tebbe, Y. Gao, M. Sastre-Rienietz, and A. Zell. A table tennis robot system using an industrial kuka robot arm. *GCPR*, 2018.

[49] Y. Gao, J. Tebbe, J. Krismer, and A. Zell. Markerless racket pose detection and stroke classification based on stereo vision for table tennis robots. *IEEE Robotic Computing*, 2019.

[50] J. Tebbe, L. Krauch, Y. Gao, and A. Zell. Sample-efficient reinforcement learning in robotic table tennis. *ICRA*, 2021.

[51] W. Gao, L. Graesser, K. Choromanski, X. Song, N. Lazic, P. Sanketi, V. Sindhwani, and N. Jaitly. Robotic table tennis with model-free reinforcement learning. *IROS*, 2020.

[52] L. Chen, R. R. Paleja, and M. C. Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. *CoRL*, 2020.

[53] L. Chen, R. R. Paleja, M. Ghuy, and M. C. Gombolay. Joint goal and strategy inference across heterogeneous demonstrators via reward network distillation. *CoRR*, abs/2001.00503, 2020.

[54] Z. Yu, Y. Liu, Q. Huang, X. Chen, W. Zhang, J. Li, G. Ma, L. Meng, T. Li, and W. Zhang. Design of a humanoid ping-pong player robot with redundant joints. *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 911–916, 2013.

[55] A. Aly, S. Griffiths, and F. Stramandinoli. Metrics and benchmarks in human-robot interaction: Recent advances in cognitive robotics. *Cognitive Systems Research*, 43:313–323, 2017.

[56] M. Huber, H. Radrich, C. Wendt, M. Rickert, A. Knoll, T. Brandt, and S. Glasauer. Evaluation of a novel biologically inspired trajectory generator in human-robot interaction. In *RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 639–644. IEEE, 2009.

[57] A. Silva, K. Metcalf, N. Apostoloff, and B.-J. Theobald. Fedembed: Personalized private federated learning, 2022. URL https://arxiv.org/abs/2202.09472.

[58] R. Paleja, M. Ghuy, N. Ranawaka Arachchige, R. Jensen, and M. Gombolay. The utility of explainable ai in ad hoc human-machine teaming. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 610–623. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/05d74c48b5b30514d8e9bd60320fc8f6-Paper.pdf.

[59] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[60] K. Choromanski, M. Rowland, V. Sindhwani, R. E. Turner, and A. Weller. Structured Evolution with Compact Architectures for Scalable Policy Optimization. In *Proceedings of the 35th International Conference on Machine Learning*, pages 969–977. PMLR, 2018.

[61] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, and J. Schmidhuber. Natural evolution strategies, 2011.

[62] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv:1703.03864*, 2017.

[63] Y. Nesterov and V. Spokoiny. Random gradient-free minimization of convex functions. *FoCM*, 2017.

[64] H. Mania, A. Guy, and B. Recht. Simple random search provides a competitive approach to reinforcement learning. *NeurIPS*, 2018.

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

[66] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7: 308–313, 1965.

[67] E. Schubert, J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Transactions on Database Systems*, (3), 2017.

[68] P. Blank, B. H. Groh, and B. M. Eskofier. Ball speed and spin estimation in table tennis using a racket-mounted inertial sensor. In S. C. Lee, L. Takayama, K. N. Truong, J. Healey, and T. Ploetz, editors, *ISWC*, pages 2–9. ACM, 2017. ISBN 978-1-4503-5188-1. URL http://dblp.uni-trier.de/db/conf/iswc/iswc2017.html#BlankGE17.

[69] Y. Gao, J. Tebbe, and A. Zell. Optimal stroke learning with policy gradient approach for robotic table tennis. *CoRR*, abs/2109.03100, 2021. URL https://arxiv.org/abs/2109.03100.

[70] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[71] Abb application manual - externally guided motion., Vasteras, 2020.

[72] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.