

A Author Contributions

- **Saminda Abeyruwan** co-led the project, developed i-S2R, ran initial i-S2R experiments, helped build and maintain robotic infrastructure, introduced a set of additional rewards to help with fine-tuning, helped with analyzing the sim-to-real discrepancies, organized the human training and evaluation protocols, and helped to write parts of the paper related to human behavior modeling via ball trajectories. Was one of the test subjects.
- **Laura Graesser** co-led the project, ran initial i-S2R experiments, advised on experimental design, helped build and maintain robotic infrastructure, analyzed the results, wrote the paper.
- **David B. D’Ambrosio** helped build and maintain robotic infrastructure and vision system. Analysis of human behavior parameters. Literature review. Post-hoc evaluation of human-robot rallies. Discussion and writing of paper.
- **Avi Singh** wrote the introduction and helped craft the overall narrative for the paper. Made the project website. Was one of the test subjects.
- **Anish Shankar** worked on the system’s hardware and software implementation, data infrastructure used in analysis and overall system performance. Was one of the test subjects.
- **Alex Bewley** helped build the vision system, curate ball detection data and trained the perception module used in this work. Contributed to writing the vision related sections of the paper.
- **Deepali Jain** co-developed with Krzysztof Choromanski: (a) the ES algorithm used to learn policies presented in this paper and (b) distributed optimization infrastructure to apply ES-based training. Conducted extensive tests of ES in the simulator in the first phase of the project. Edited the paper.
- **Krzysztof Choromanski** co-developed with Deepali Jain: (a) the ES algorithm used to learn policies presented in this paper and (b) distributed optimization infrastructure to apply ES-based training. Conducted extensive tests of ES in the simulator in the first phase of the project. Edited the paper.
- **Pannag R. Sanketi** managed the team. Set the research direction, co-led the project, and wrote the paper. Was one of the test subjects.

B Details on the BGS Algorithm

As described in [section 3](#), the ES objective is given by:

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

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 .

ES does not use derivatives or back-propagation to update policy parameters. Instead, the gradient of the policy parameters θ with respect to the objective is estimated with various Monte Carlo techniques. In this work we apply Monte Carlo leveraging in addition the antithetic sampling trick, widely applied by the community.

Specifically, θ is perturbed either by adding or subtracting Gaussian perturbations δ_{R_i} and completing environment rollouts using the perturbed parameters. As a result each perturbation is associated with a reward, one for each direction R_i^+ and R_i^- .

Assuming the perturbations, δ_{R_i} , are rank ordered with δ_{R_1} being the top performing direction, then the policy update can therefore expressed as follows.

$$\theta' = \theta + \alpha \frac{1}{\sigma^R} \sum_{i=1}^k \left[\left(\left(\frac{1}{m} \sum_{j=1}^m R_{i,j}^+ \right) - \left(\frac{1}{m} \sum_{j=1}^m R_{i,j}^- \right) \right) \delta_{R_i} \right], \quad (3)$$

where α is the step size, σ^R is the standard deviation of each distinct reward (positive and negative direction), k is the number of top directions (elites), N is the number of directions sampled per parameter update, and $k < N$. m is the number of repeats per direction and $R_{i,j}^+$ is the reward

corresponding to the j -th repeat of i -th in the positive direction. $R_{i,j}^-$ is the same but in the negative direction.

Our BGS algorithm is built on two pillars that we describe in detail below.

Novel Elite-Choice Algorithm: One of the key features of BGS is the novel algorithm of selecting top directions (the *elites*). In ARS [68], the ranking of the elites is determined by treating each antithetic direction separately. All rewards are ranked yielding an ordering of directions based on the absolute rewards of either the positive or negative directions (Equation 4). Whereas in BGS we take the difference in rewards between the positive and negative directions and rank the differences to yield an ordering over directions (Equation 5).

$$ARS : \text{Sort } \delta_{R_i} \text{ by } \max\{R_1^+, \dots, R_i^+, R_1^-, \dots, R_i^-\}. \quad (4)$$

$$BGS : \text{Sort } \delta_{R_i} \text{ by } \max\left\{\left(\frac{1}{m} \sum_{j=1}^m R_{1,j}^+ - \frac{1}{m} \sum_{j=1}^m R_{1,j}^-\right), \dots, \left(\frac{1}{m} \sum_{j=1}^m R_{i,j}^+ - \frac{1}{m} \sum_{j=1}^m R_{i,j}^-\right)\right\}. \quad (5)$$

ARS can be interpreted as ranking directions in absolute reward space, whereas BGS ranks directions according to reward curvature because it ranks based on reward deltas. The new elite-choice algorithm was the game changer for all policy-training experiments. We could not train efficient policies with ARS (even in the simulator).

Orthogonal Perturbations (Samples): The other key feature of the BGS algorithm is the use of the orthogonal ensembles of perturbations (samples) δ_{R_i} . This technique was originally introduced in [64] and relies on constructing perturbations δ_{R_i} in blocks, where each block consists of pairwise orthogonal samples. Those samples are still of Gaussian marginal distributions, matching those of the regular non-orthogonal variant. The feasibility of such a construction comes from the isotropic property of the Gaussian distribution (see: [64] for details). We observed that orthogonal perturbations led to faster convergence in training.

In addition to our novel-elite choice algorithm and orthogonal perturbations, we apply a number of common approaches used in ES methods; state normalization [66, 75], reward normalization [68], and perturbation filtering [66]. We also repeat and average rollouts with the same parameters to reduce variance.

C Iterative-Sim-to-Real Procedure

For the table tennis rallying task, we found 3 iterations to be sufficient. The policy was trained for 30k to 45k updates for the first round of training in simulation since it has to learn everything from scratch. For subsequent simulation rounds, the policy was only trained for 5k updates, since we warm start from latest real world policy weights and its primary task here is *adaptation* to a change in human behavior. Due to the human cost of real world fine-tuning and evaluation, we did not experiment with shorter or longer training cycles. In the real world, the policy was fine-tuned for 70 parameter updates per cycle for the last two cycles and 60 updates for the first cycle to make 200 updates in total. This is equivalent to approximately 2 hours of wall clock time per cycle, which was our budget per player.

C.1 Seed Selection for Rounds of Simulated Training

We have used the following methodology when training in simulation. When training in simulation is required, whether it is training from scratch or intermediate steps of i-S2R, we train 3 models with 3 different random seeds. Different random seeds were used for different players. When transferring to the physical robot, each model is evaluated for 50 episodes according to the training and evaluation instructions provided in subsection H.1. The model with the highest average return is selected for fine-tuning and further experiments. We have used a simple, sparse reward structure for evaluation: if the robot hits the ball, a reward of +1 is given, and if the ball lands on the human side, an additional +1 is given reward. Therefore, the maximum episode reward is +2. If the robot misses the ball, there is no reward, and if the robot faulted or stopped during an episode, a -2 reward is assigned to the episode.

C.2 Bringing the Fine-Tuned Model from the Real World Back to Simulation

In i-S2R the fine-tuned model from the real world is brought back to simulation in the next iteration for two reasons. First, the fine-tuned model has been trained on the most up to date human behavior. As a result it is likely better adapted to play with the updated human behavior model than the latest set of policy weights from simulated training which were trained on the prior human behavior model.

Second, there are aspects of our real world system which we have not been able to model accurately in simulation on top of the challenges in modeling human behavior. Our vision system does not detect spin, there are calibration defects, variability in estimated delays, conditions of the surface materials, wear and tear (of table tennis balls), physical robot properties mismatched with the simulated robot. Therefore, our policies are subject to a sim2real gap and real world fine-tuning adapts the policy to real world conditions. When we transfer the fine-tuned policy weights back to simulation we observe that some adaptation to real world conditions persist from iteration to iteration, reducing the adaptation time in subsequent fine-tuning iterations.

However it is possible that this approach makes training in simulation more difficult. It would be interesting to compare our approach with a variant in which the policy weights are not transferred back to simulation from the real world. Instead training in simulation would continue using the latest policy weights from the previous iteration but using the latest human model after real fine-tuning. We leave this to future work.

C.3 Human Behavior Models

Table 1 shows the changes of the ball behavior models, M_0 , M_1 , and M_2 , for each player. Skill levels: players 3 and 5 are beginners, players 2 and 4 are intermediate, and player 1 is advanced.

	player 1			player 2			player 3			player 4			player 5		
	M_0	M_1	M_2												
min z velocity (ms^{-1})	1.25	-1.47	-1.56	0.88	-1.14	-1.27	0.64	-1.23	-1.39	0.04	-1.31	-1.72	0.52	-0.70	-0.87
max z velocity (ms^{-1})	2.71	2.95	2.95	2.84	2.84	3.07	2.49	2.79	2.79	2.25	2.73	2.73	2.59	2.75	2.75
max x velocity ($ ms^{-1} $)	1.70	3.05	3.05	1.41	2.81	2.89	0.79	2.59	2.78	1.50	3.40	3.45	0.68	2.30	2.68
min y velocity ($ ms^{-1} $)	4.12	2.17	2.17	3.97	3.52	2.74	2.95	2.19	2.19	4.44	3.33	2.96	4.20	2.73	2.70
max y velocity ($ ms^{-1} $)	6.31	6.63	6.63	6.38	8.05	8.82	6.03	7.11	7.11	7.33	7.33	7.36	6.34	6.94	6.94
x start min (m)	-0.19	-0.79	-0.83	0.02	-0.86	-0.87	0.10	-0.93	-0.93	-0.09	-0.8	-0.83	0.25	-0.64	-0.80
x start max (m)	0.19	0.63	0.70	0.73	0.65	0.67	0.61	0.79	0.81	0.68	0.78	0.83	0.42	0.55	0.64
y start min (m)	1.05	0.04	0.04	0.85	0.37	0.08	0.61	0.05	0.04	1.01	0.21	0.04	1.08	0.17	0.17
y start max (m)	2.51	1.87	1.92	1.68	1.89	1.95	1.35	1.83	1.92	1.88	1.58	1.58	1.44	1.81	1.82
z start min (m)	0.07	0.19	0.19	0.15	0.08	0.01	0.18	-0.15	-0.29	0.15	0.24	0.19	0.33	0.26	0.26
z start max (m)	0.62	0.83	0.83	0.45	0.59	0.63	0.52	1.10	1.11	0.72	0.72	0.76	0.58	0.76	0.79
x land min (m)	-0.01	-0.62	-0.71	-0.08	-0.52	-0.68	-0.08	-0.67	-0.74	-0.02	-0.19	-0.63	0.07	-0.66	-0.666
x land max (m)	0.67	0.74	0.74	0.76	0.76	0.76	0.76	0.76	0.76	0.75	0.75	0.76	0.58	0.73	0.73
y land min (m)	-1.35	-1.37	-1.37	-1.34	-1.36	-1.37	-1.33	-1.37	-1.37	-1.37	-1.37	-1.37	-1.31	-1.37	-1.37
y land max (m)	-0.2	-0.15	-0.15	-0.23	-0.15	-0.15	-0.22	-0.18	-0.16	-0.27	-0.21	-0.15	-0.30	-0.16	-0.16

Table 1: Ball distribution changes, M_0 , M_1 , and M_2 , per player for i-S2R.

C.4 Details on Modeling Human Ball Distributions

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.

D Hardware Details

D.1 Robot Hardware Overview

Player Robot: The player robot (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. The robot arm’s end effector is a standard table tennis paddle with the handle removed attached to a 174.3mm extension. The arm is controlled with ABB’s Externally Guided Motion (EGM) interface at approximately 248Hz by specifying joint position and speed targets [76]. The 2D linear actuator is independently controlled at up to 125Hz with position target commands for each axis at a fixed velocity through Festo’s custom Modbus interface. Position feedback from the robots is received at the command rate. The policy outputs individual joint velocity commands which are converted by a safety layer (to prevent collisions / stay within performance limits) into raw hardware commands. The robot starts from a forehand-pose as the home position and is controlled by the learned policy as soon as a ball is in play. As soon as the policy either makes contact with the ball returning it or misses it, the robot is returned to the home position and continues the rally with the next or returned ball as fresh inputs to the policy.

D.2 Ball Vision Model

The ball location is determined through a stereo pair of Ximea MQ013CG-ON cameras positioned above and to the side of the table and running at 125Hz. A recurrent 2D detector model detects the ball position in each camera independently. This detector was trained with ≈ 2 hours of ball video data with an additional ≈ 15 minutes of humans pretending to play without a ball which is used for hard negative mining. During training, horizontal flipping augmentation are applied to video sequences to balance detection performance across both directions. The 2D detections from each camera are fed to standard OpenCV triangulation to produce 3D coordinates, which are in turn run filtered through a 3D tracker and interpolated to the 75Hz frequency that the policy does inference on. There is roughly $\approx 15ms$ of lag between image capture and 3D coordinate availability.

E Model Architecture

We represent our policy using a three layer 1D fully convolutional gated dilated CNN with 976 parameters. Details are given in Table 2. The observation space is 2-dimensional (timesteps x [ball position, robot joint position]) which is an (8 x 11) matrix. The networks outputs a vector (8,) representing joint velocities.

Parameter	Layer		
	1	2	3
Convolution dimension	1D	1D	1D
Number of filters	8	12	8
Stride	1	1	1
Dilation	1	2	4
Activation function	tanh	tanh	tanh
Padding	valid	valid	valid

Table 2: CNN model architecture.

F Training Hyperparameters

Table 3 presents the ES hyper-parameters used for both simulated and real world training.

Parameter	Simulation	Real fine-tuning
Step size	0.00375	0.00375
Perturbation standard deviation	0.025	0.025
Number of perturbations	200	5
Number of rollouts per perturbation	15	3
Percentage to keep (top x% rollouts)	30%	60%
Maximum environment steps per rollout	200	200
Use orthogonal perturbations	True	True
Use observation normalization	True	True

Table 3: ES hyperparameters.

G Simulation details

Our simulation handles robot dynamics and contact dynamics (via PyBullet), and we model the ball using Newtonian dynamics, incorporating air drag but not spin. At the beginning of an episode, a ball throw is sampled according to the parameterized distribution described in section 4.

One major difference between simulated and real world robotic systems is the existence of sensor latency and noise in the latter but not the former. We seek to minimize this difference by measuring the latency of the major system components and modeling them in our simulation. These components include **(a)** ABB and Festo action latency, **(b)** ball observation latency, **(c)** ABB and Festo observation latency. The latency of each component is modeled by $\mathcal{N}(\mu, \sigma^2)$ where μ and σ^2 were measured empirically. The details are given in subsection G.1. At the beginning of each episode during training in simulation the latency of each component is sampled and remains fixed throughout the episode.

G.1 Sensor Latency Model

Table 4 details the parameters used in the simulated sensor latency model described above.

Component	Latencies (ms)	
	μ	σ^2
Ball observation	40	8.2
ABB observation	29	8.2
Festo observation	33	9
ABB action	71	5.7
Festo action	64.5	11.5

Table 4: Sensor latency model parameters per component.

G.2 Rewards in Simulation and the Real World

Table 5 describes the rewards used in simulation to train and fine-tune in the real world. Rewards 1 - 3 are common between simulation and the real world. The fault reward (4) is only available on a physical robot. Rewards 6 - 8 are proxies for this in simulation. Rewards 9 - 11 are used in simulation to encourage the policy to learn safe style (e.g. paddle not coming close to the table) to reduce the likelihood of collisions in the real world upon transfer.

Reward	Range	Sim weight	Real weight	Sim weighted max score	Real weighted max score
(1) State transition plus bonus for landing the ball close to a target in the center of the table	[0, 5]	1	1	5	5
(2) Bonus for clearing the net with a target height	[0, 1]	1	1	1	1
(3) Bonus for hitting the ball and landing it on the opponent side of the table	[0, 1]	0.1	0.1	0.1	0.1
(4) Actual fault reward in real	{-2, 0}	0.0	1.0	0	0
(5) Episodic jerk reward (proxy for faulting in real)	[0, 1]	0.3	0	0.3	0
(6) Episodic acceleration reward (proxy for faulting in real)	[0, 1]	0.3	0	0.3	0
(7) Episodic velocity reward (proxy for faulting in real)	[0, 1]	0.4	0	0.4	0
(8) Episodic joint angle reward (safety reward, aimed to prevent faulting in real)	[0, 1]	1	0	1	0
(9) Safety reward, penalty for robot colliding with itself or table	[-1 * timesteps, 0]	1	0	0	0
(10) Paddle height reward	[-1 * timesteps, 0]	0.5	0	0	0
(11) Style reward (sim only)	[-1 * timesteps, 0]	1	0	0	0
Total				8.1	6.1

Table 5: Rewards used in simulation to train and fine-tune in the real world.

H Evaluation Methodology

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) 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*”).

We introduced a bijective model for anonymization to make it easier for the players to evaluate the models fairly. Each player evaluated all their ten models and three models trained by two other randomly selected players in the roster. The identity of the models is revealed once all the evaluations have been completed. A successful evaluation must contain at least 50 valid rally balls (see subsection H.2 for further instruction on determining a valid rally ball). In addition to rally length, we have also collected statistics such as whether the player or robot is at fault for ending the rally.

All players trained and evaluated the following ten models:

1. i-S2R sim 1

2. i-S2R fine-tuned 35%
3. i-S2R sim 2
4. i-S2R fine-tuned 65%
5. i-S2R sim 3
6. i-S2R fine-tuned 100%
7. S2R fine-tuned 65%
8. S2R fine-tuned 100%
9. S2R-Oracle sim
10. S2R-Oracle fine-tuned

Each player cross evaluated three models each from two other players:

1. i-S2R fine-tuned 100%
2. S2R-Oracle fine-tuned
3. S2R fine-tuned 100%

Table 6 shows the trainer and evaluator combinations for cross evaluations.

Trainer	Evaluators	
player 1	player 4	player 5
player 2	player 1	player 3
player 3	player 1	player 4
player 4	player 2	player 5
player 5	player 2	player 3

Table 6: Trainer and evaluator combinations.

H.1 Instructions for Human Players

We have provided the following instructions while gathering initial ball trajectories and rallying with the robot.

Initial Ball Distribution: The player lobs the ball over the net from the left hand quadrant of the opponent side to the right hand quadrant of the robot side. All the players used the same standard table tennis racket.

Training and Evaluation: The player always starts a rally. The player lobs the ball from the left hand quadrant of the opponent side to the right hand quadrant of the robot side as naturally as possible. During the play, for all the return balls from the robot, the player tries to return the ball to the right hand quadrant of the robot. In all cases, we have instructed the player to cooperate with robot as much as possible.

H.2 Details on Rally Score Evaluations

Table 7 contains the rally length evaluation and end-of-rally attribution instructions for raters. For each evaluation, the cases marked as “Filter” are removed. Then, the top 60 rallies are selected and sorted by rally length. For reporting, we have selected the top 50 rallies from this set.

Description	did-robot-end-rally	Instruction
Human hit the first ball to the net.	-	Filter
Human hit the first ball over the table.	-	Filter
Human hit the first ball out of distribution, robot did not return.	Yes	
Human hit the first valid ball and the robot did not react.		Filter
Human returning a ball out of distribution, robot did not return.	Yes	
Human returns a ball that bounces multiple times on the human side (robot has returned the ball).	No	
Human returning a ball over the table.	No	
Human returning a ball to the edge of the table, robot did not return.	Yes	
Human hit a ball that graces the net which robot did not contact.	-	Filter
Human hit a ping pong type service and the robot did not return.	-	Filter
Robot returns a ball which graces the net, but the human cannot return.	No	
Robot returns a ball which lands at the corner of the table and the human cannot return.	No	
Rally ends due to the robot cannot contact the ball and/or the encoder diff is high (obvious behavior change from a previous rally, if applicable)	Yes	
Robot is in ABB home pose, not the episode start state. You throw a dummy ball by hand so that the robot moves to episode start state.	-	Filter
Robot is in ABB home pose, not the episode start state. You throw with a paddle so that the robot moves to episode start state.	-	Filter

Table 7: Rally score evaluation and end-of-rally attribution.

I Player Skill Level

Table 8 contains further details on player rally length, calculated over all 10 models that a player evaluated (see Appendix H). This data was used to group players into three skill levels; beginner (players 3 and 5), intermediate (players 4 and 2), and advanced (player 1). Note that player 5 was the non author player.

We grouped players according to empirical skill (i.e. how they actually played) as opposed to using self-reported skill because non-professional players’ perception of their skill level may not be well calibrated across players. In future it would be interesting to consider self-reported skill in addition to empirical skill.

J Rally Length Normalization Details

Let x be the rally length, μ_x the mean rally length, and σ_x the standard deviation of the rally lengths, then rally length is normalized as follows:

$$\frac{x - \mu_x}{\sigma_x} \tag{6}$$

Evaluations Here μ_x and σ_x are calculated over all 10 evaluations (see Appendix H), making 500 (10 x 50) rallies in total. The values per player are given in Table 8. This can be interpreted

Player	Rally length				
	Min	25th	Mean (Std.)	75th	Max
3 (beginner)	2	3.0	7.0 (5.9)	9.0	52
5 (beginner)	2	3.8	10.4 (10.9)	13.0	85
4 (intermediate)	2	4.0	14.0 (16.1)	18.0	117
2 (intermediate)	2	4.0	16.8 (27.1)	15.0	190
1 (advanced)	2	5.0	19.4 (27.6)	22.0	345

Table 8: Rally length statistics by player. Values were calculated over all 10 models that a player evaluated (see Appendix H), making 500 (10 x 50) rallies in total

as normalizing for player skill and is intended to make rally length comparable between players of different skill levels (e.g. beginner, advanced). This approach was used in Figure 3 and Figure 8.

Cross-Evaluations Here μ_x and σ_x are calculated per model (50 rallies in total) and rallies are normalized with respect to the player who trained the model. This is intended to make rally length comparable across models and players (e.g. S2R+FT player 3, i-S2R player 1). This approach was used in Figure 7 and Figure 12 to estimate the % difference in performance when a model is evaluated by different players (cross-evaluations) who did not train the model.

K Additional Results

Here we present additional results. Figure 8 contains additional presentations of the data aggregated over all five players; (a) mean normalized rally length, (b) distribution of normalized rally length, and (c) mean rally length. Figure 9 presents mean rally length by player skill level. Figure 10 contains additional presentations of the data aggregated over 4/5 players with the outlier (advanced player) excluded; (a) mean normalized rally length, (b) distribution of normalized rally length, and (c) mean rally length.

Figure 11 and Figure 12 break out results per player. Note that player 5 was the non-author player and was categorized as a beginner. Figure 11 shows the mean and distribution of rally length for each player, ordered from top to bottom by skill level, beginner to advanced. Figure 12 shows cross evaluation data by player with the same ordering by player skill.

Figure 13, Figure 14, and Figure 15 present additional details on ball distributions per player during training and evaluation.

Figure 16 and Figure 17 present additional data on the robot return rate per player in the form of heatmaps. The color of each square represents the robot return rate (darker = higher return rate) and the number in each square represents the percentage of balls. The grid operates on two scales, a large 3 x 3 grid, and within each cell, a smaller 3 x 3 grid. In each heatmap, the large scale grid represents where the incoming ball bounced on the robot side of the table.

In Figure 16 the small scale grid represents the position on the player side where the ball originated. So, Figure 16 shows the conditional return rate given the start position of the incoming ball and where the ball bounced on the robot side of the table. For example, let’s look at the player 3 i-S2R (top left grid). The middle large grid represents the middle of the robot side of the table, and shows that 48.6% of the balls land here, out of which 12.6% are coming from the opponent (human) hitting the ball from the far left of the human side of the table, and 0.3% from the middle right of the human side of the table.

In Figure 17 the small scale grid represents the position on the player side where the ball landed (i.e. where the robot returned the ball to). So, Figure 17 shows the conditional return rate given the landing position of the returned ball (i.e. where the robot hit it to) and where the incoming ball bounced on the robot side of the table. As an example, if we look at the player 3 i-S2R middle grid, it accounts for 53.4% of the balls, out of which 17.4% of the returns are to the middle of the table.

Statistical Significance We note that the un-normalized mean rally lengths for i-S2R, S2R+FT and S2R-Oracle+FT are not statistically significantly different, since the 95% confidence intervals overlap (Appendix K, Figure 8 (c)). However, the histogram of rally lengths for i-S2R and S2R+FT (Figure 3, right) shows that a large fraction of the rallies for S2R+FT are shorter (i.e. less than 5), while i-S2R achieves longer rallies more frequently. This suggests i-S2R yields policies that are more fun to play with on average.

When rally length is normalized to account for differences in skill level between players (Appendix K, Figure 8 (a)), the mean rally length for S2R-Oracle+FT is statistically significantly higher than S2R+FT, although the difference is small.

Finally, when the advanced player (outlier) is excluded (Figure 10), the mean rally length (normalized and un-normalized) for i-S2R is statistically significantly higher than S2R+FT and the difference is large. The mean rally length for i-S2R and S2R-Oracle+FT are not statistically significantly different.

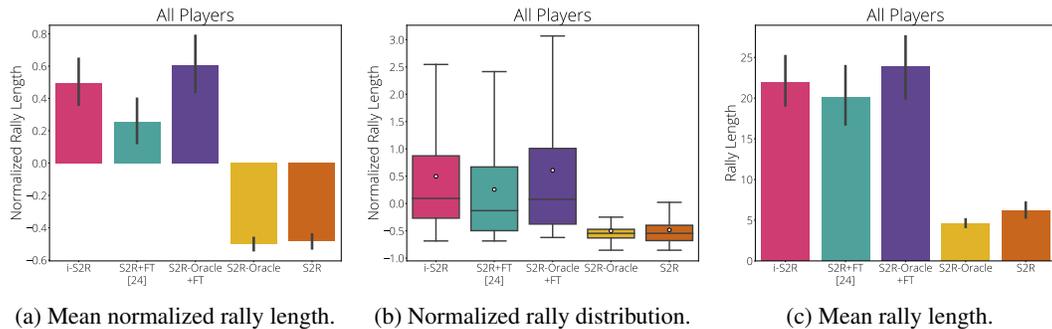


Figure 8: Aggregated results across all 5 players after learning. Vertical lines are 95% confidence intervals (CIs).

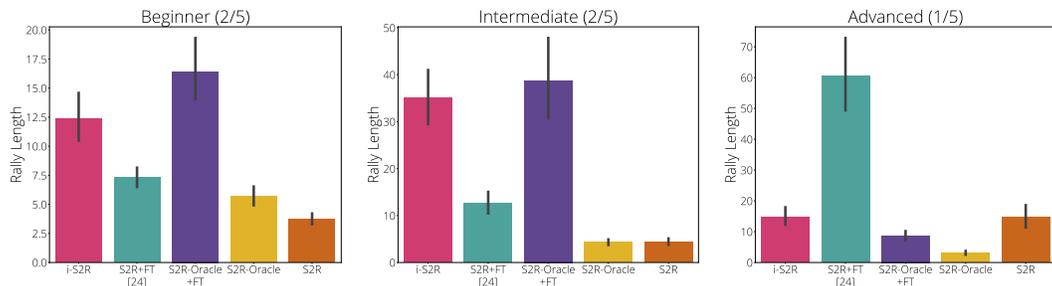


Figure 9: Mean rally length by player skill level. Vertical lines are 95% confidence intervals (CIs). *Note:* S2R-Oracle+FT is only getting 35% of i-S2R and S2R+FT fine-tuning budget.

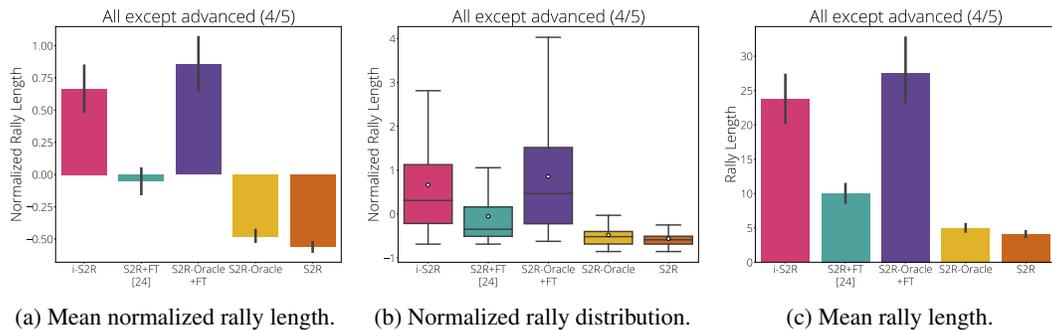


Figure 10: Aggregated results across 4/5 players, excluding the outlier (advanced) player, after learning. Vertical lines are 95% confidence intervals (CIs).

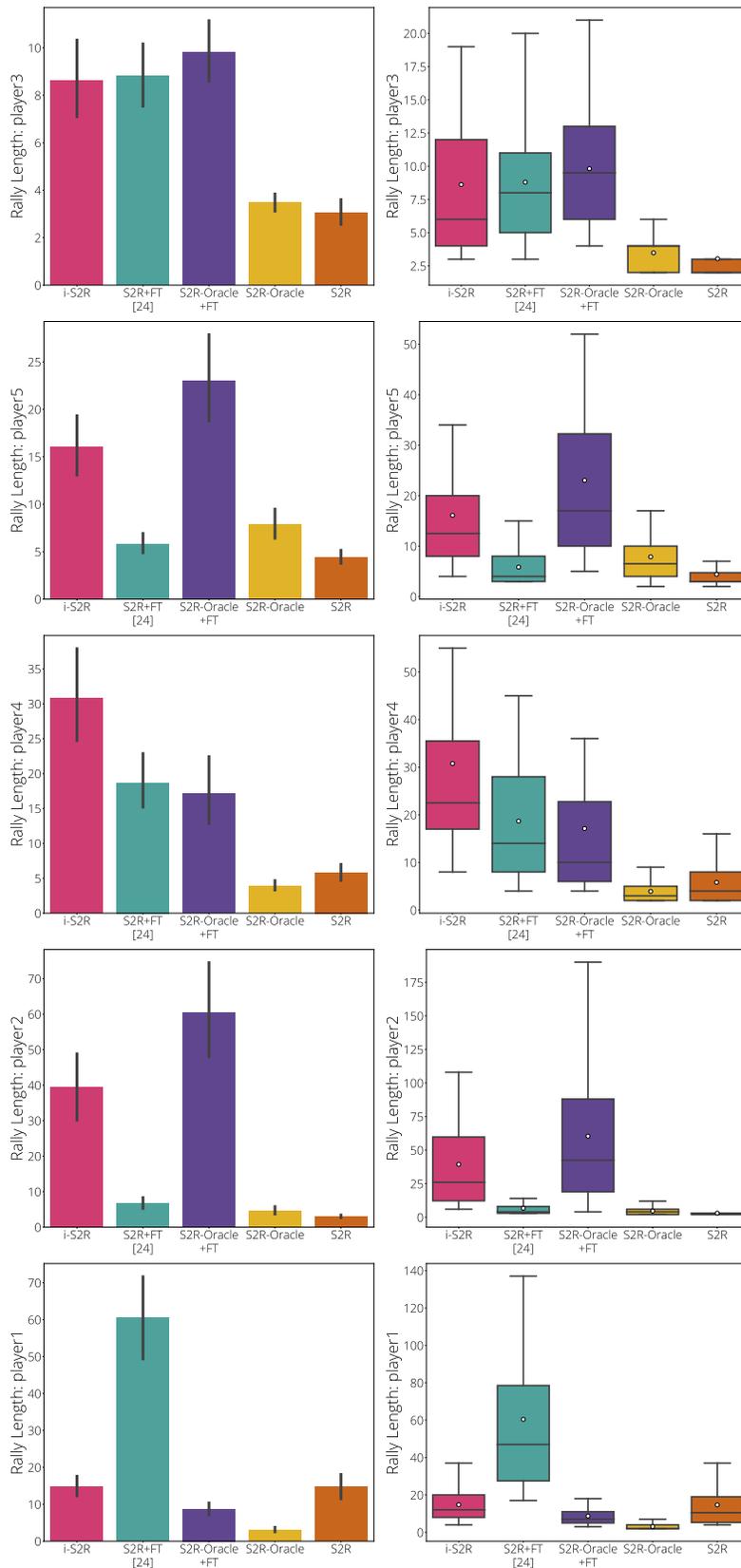


Figure 11: Breakdown by player, order top to bottom from lowest to highest overall rally mean. **left:** Mean rally length. Vertical lines are 95% CIs. **right:** Rally length distribution per model.

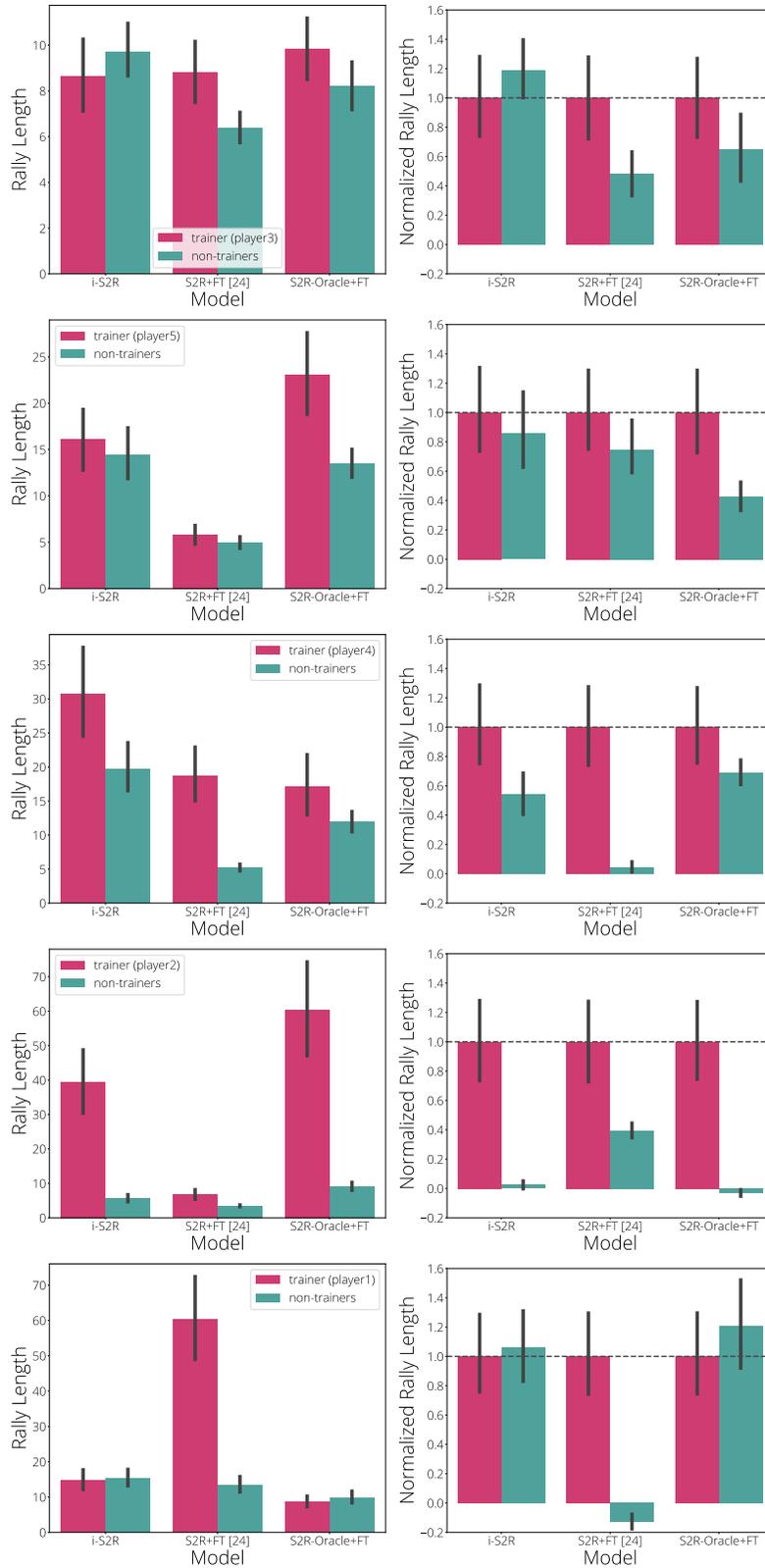


Figure 12: Cross evaluations. Ordered by model trainer, top to bottom from lowest to highest overall rally mean. **left:** Mean rally length. **right:** Mean normalized rally length. Vertical lines are 95% CIs.

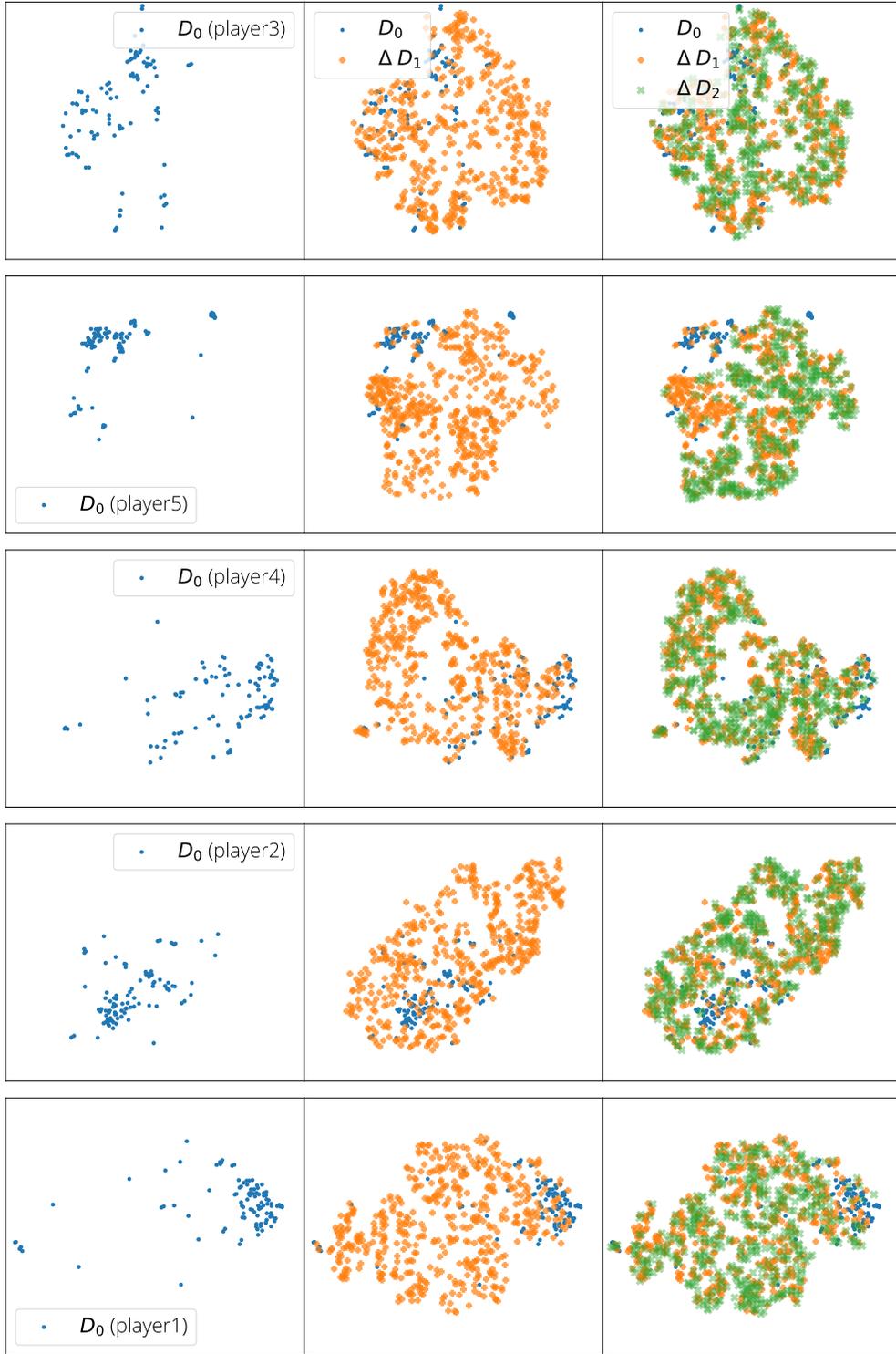


Figure 13: Evolution of ball distributions for each player projected to 2D using t-SNE [77] (up to 500 random ball trajectory are used for ΔD_1 and ΔD_2 , and $D_i = D_0 + \sum_{j=1}^i \Delta D_j$.)



Figure 14: Evolution of ball distribution for each player and the overlapping evaluation distributions for i-S2R(2D projected using t-SNE [77] and up to 500 random ball trajectory are sample from each round).

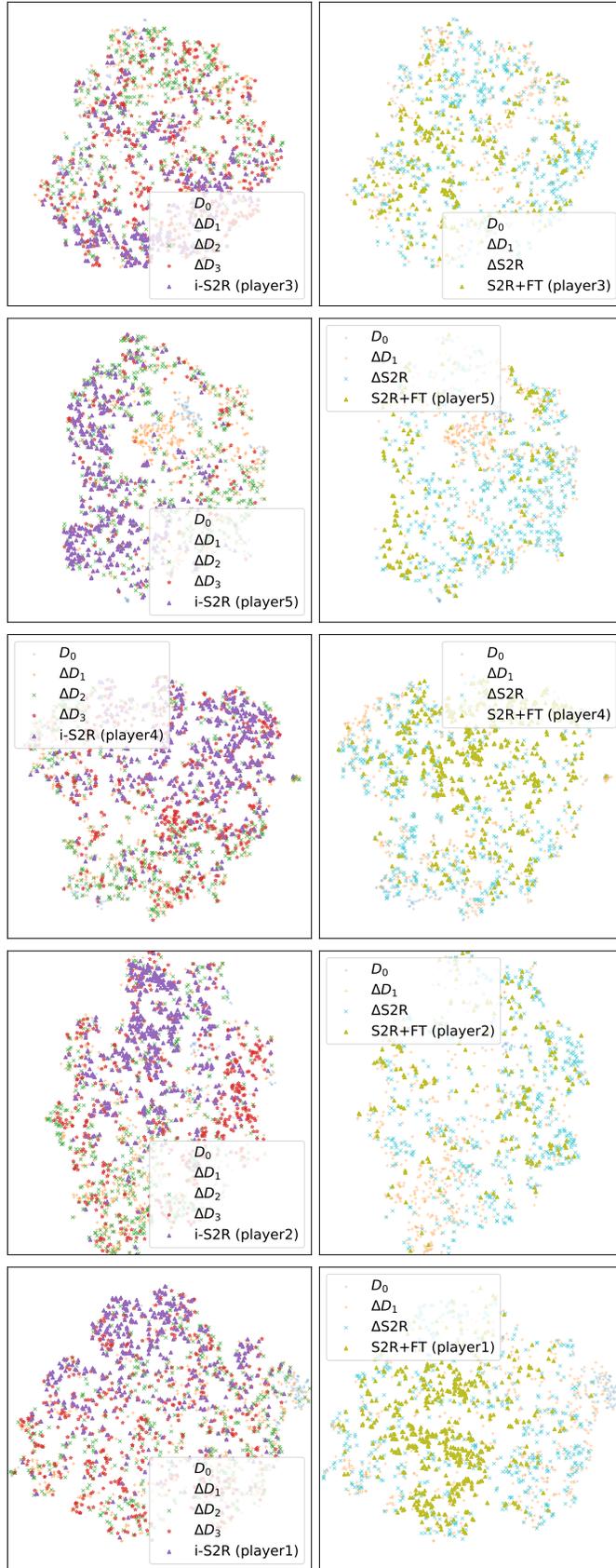


Figure 15: Evolution of ball distribution for each player and the overlapping evaluation distributions for i-S2R **left** and S2R+FT on **right** (2D projected using t-SNE [77] and up to 500 random ball trajectory are sample from each round).

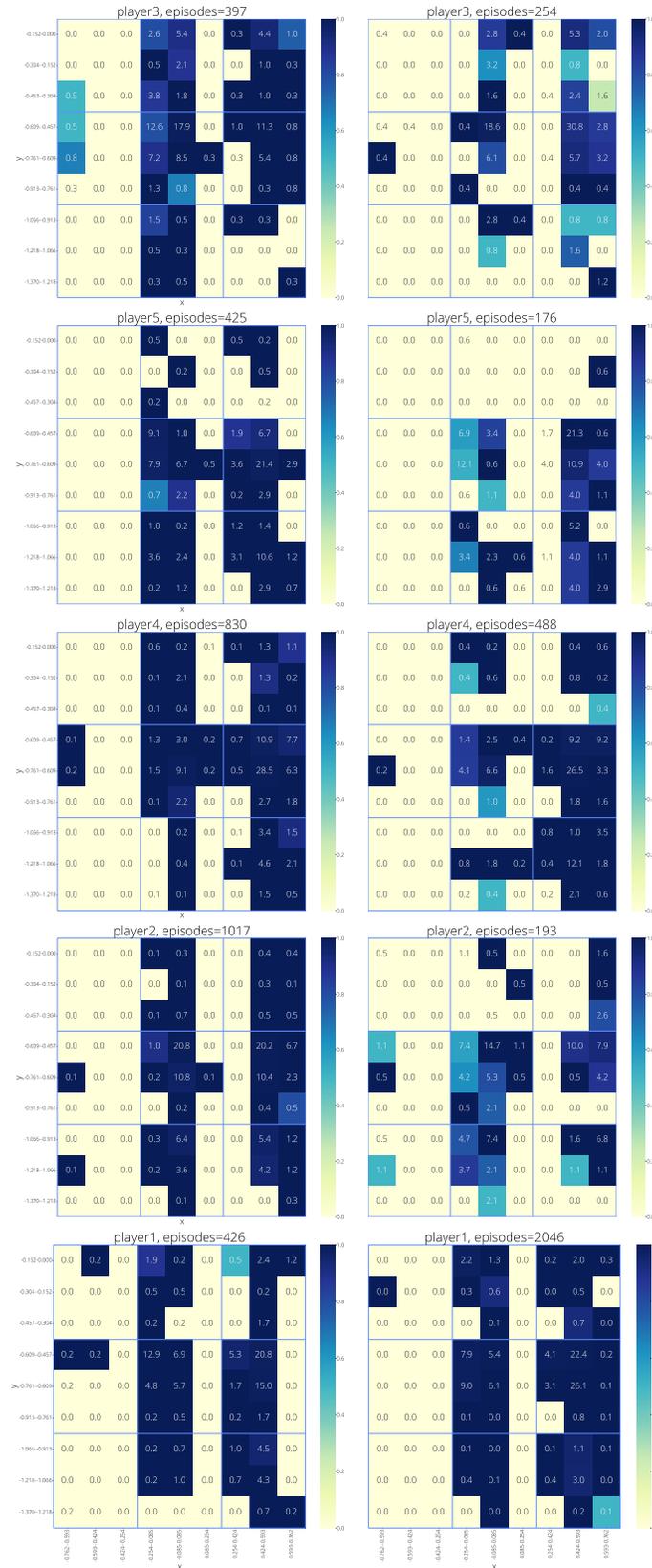


Figure 16: Heatmaps of the robot hit rate with respect to the (x, y) position where the episode initiated from. **left:** i-S2R **right:** S2R+FT. The outermost block represents the robot side. Each 3x3 blue block represents the human (opponent) side of the table. Each block shows, if the human throw landed on the robot side, where would the human throw initiated from. The block color represents the robot hit rate.

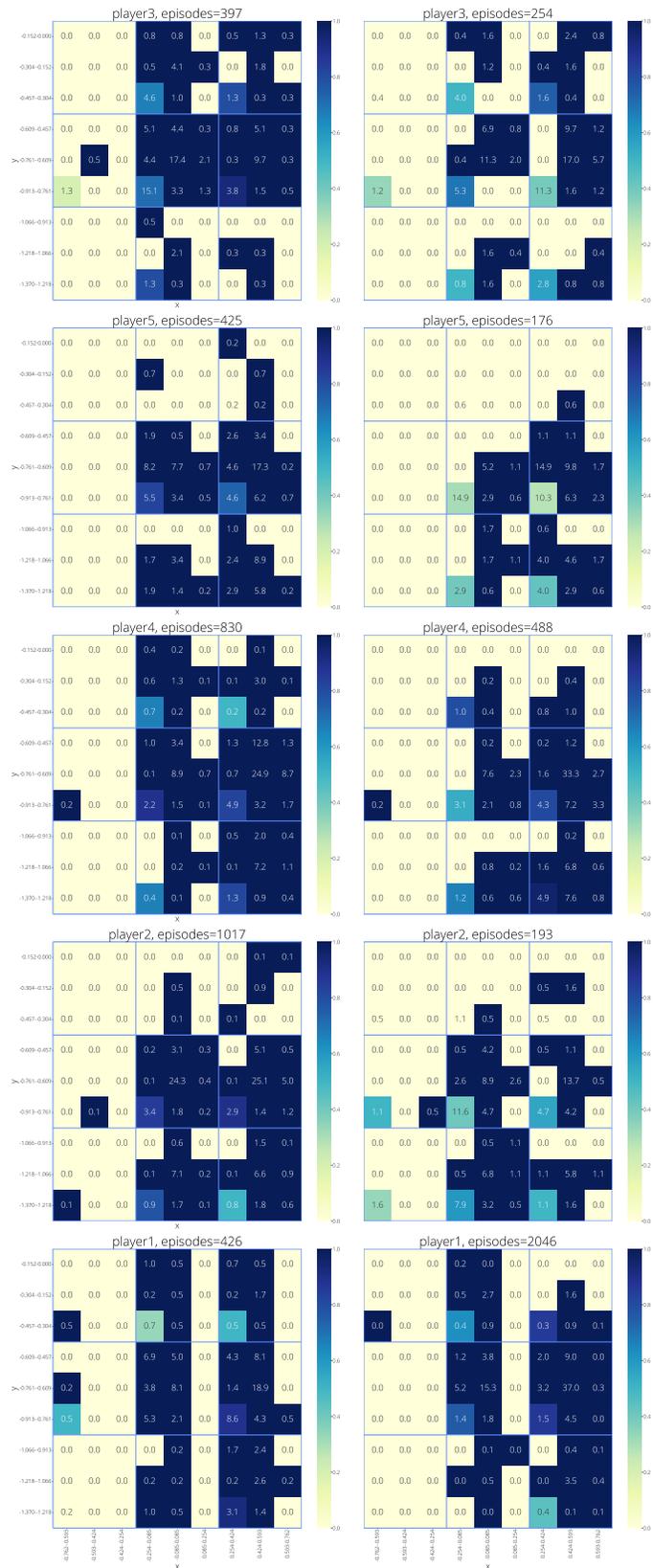


Figure 17: Heatmaps of the robot hit rate with respect to the (x, y) position where the episode ends. **left:** i-S2R **right:** S2R+FT. The outermost block represents the robot side. Each 3x3 blue block represents the human (opponent) side of the table. Each block shows, if the human throw landed on the robot side, where would the robot hit the ball such that it lands on the opponent side. The block color represents the robot hit rate.

K.1 Oracle Ball Distribution Ablation

To assess the contributions of the human behavior model, we hand-designed large, medium, and narrow ball distributions as shown in Table 9. The medium distribution is restricted to throwing balls towards the forehand hand side of the robot with a restricted velocity range, and the narrow distribution is modeled based on our ball thrower machine. Figure 18 compares these distributions with S2R-Oracle. Evaluations were done using a single human subject and are zero-shot from simulation with no fine-tuning. The large model performed $\approx 40\%$ lower than S2R-Oracle. Further, we observe that lower zero-shot scores substantially increase the fine-tuning time required to match final performance, which is costly when a human is in the loop. We also observe that zero-shot transfer with the medium distribution is comparable to S2R-Oracle. This is because there is a high overlap between the two human behavior models. This indicates that human behavior modeling via ball distributions plays an important part in the ability for cooperative interaction with a table tennis playing robot.

Parameter	Large	Medium	Narrow	S2R-Oracle
min z velocity (ms^{-1})	-10	-0.1	-1.2	-1.72
max z velocity (ms^{-1})	10	2	1.5	2.72
max x velocity ($ ms^{-1} $)	10	1.5	0.9	3.45
min y velocity ($ ms^{-1} $)	2	3.5	5.0	2.96
max y velocity ($ ms^{-1} $)	35	8.5	9.4	7.35
x start min (m)	-0.76	-0.75	0.15	-0.82
x start max (m)	0.76	0.4	0.55	0.82
y start min (m)	0.1	1.2	1.01	0.03
y start max (m)	2.0	1.37	1.57	1.58
z start min (m)	-0.4	0.15	0.25	0.19
z start max (m)	1.2	0.6	0.64	0.75
x land min (m)	-0.76	-0.2	0.18	-0.62
x land max (m)	0.76	0.7	0.62	0.75
y land min (m)	-1.37	-1.3	-1.26	-1.36
y land max (m)	-0.1	-0.5	-0.33	-0.15

Table 9: The ball distribution parameters for each of the ablated distributions. *land* here implies landing on the robot side.

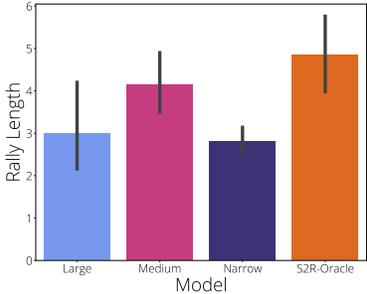


Figure 18: Zero-shot transfer of rally length for different ball distributions as defined in Table 9.