Robot Air Hockey: A Manipulation Testbed for Robot Learning with Reinforcement Learning

Caleb Chuck[1,*], Carl Qi[1,*], Michael J. Munje[1,*], Shuozhe Li[1,*], Max Rudolph[1,*], Chang Shi[1,*], Siddhant Agarwal[1,*], Harshit Sikchi[1,*], Abhinav Peri[1], Sarthak Dayal[1], Evan Kuo[1], Kavan Mehta[1], Anthony Wang[1], Peter Stone[1,3], Amy Zhang[1], Scott Niekum[2]

[1] The University of Texas at Austin
[2] University of Massachusetts Amherst
[3] Sony AI

***Abstract*—Reinforcement Learning is a promising tool for learning complex policies even in fast-moving and object-interactive domains where human teleoperation or hard-coded policies might fail. To effectively reflect this challenging category of tasks, we introduce a dynamic, interactive RL testbed based on robot air hockey. By augmenting air hockey with a large family of tasks ranging from easy to challenging, goal-conditioned or multiobject, our testbed allows a varied assessment of RL capabilities. The robot air hockey testbed also supports sim-to-real transfer with three domains: two simulators of increasing fidelity and a real robot system. Using a dataset of demonstration data gathered through two teleoperation systems: a virtualized control environment, and human shadowing, we assess the testbed with behavior cloning, offline RL, and RL from scratch.**

***Index Terms*—Reinforcement Learning, Dynamic robotic manipulation, Skill learning.**

I. Introduction

Reinforcement Learning (RL) offers a promising direction for real-world robotics by allowing robotics to accomplish complex tasks using only a reward description. Although high-performance demonstrations can sometimes be collected through tools like teleoperation, it is not always feasible (and often not possible) to collect large quantities of such "expert" data. Recent developments in RL [1]–[3] offer ever-increasing generalization capabilities, and improved learning algorithms through goal-conditioned RL [4]–[6] or unsupervised skill learning [7]–[11], as well as utilization of human information such as preferences and offline demonstrations [12]–[16]. Suitable testbeds for assessing RL in the real world will help scale RL to real-world robotics.

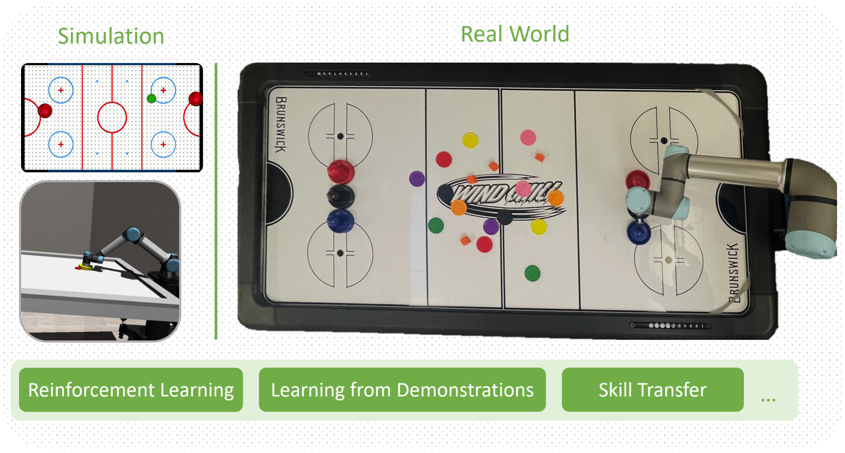
Real-world environments are challenging because they are often both dynamic and interactive. For example, objects can roll about the table when cooking or might fall when cleaning. The world is full of *dynamic elements*—state features that are constantly moving. By contrast, *quasistatic elements* remain predominantly stationary and only move when being operated on by the robot. Furthermore, real-world tasks often require the agent to *interact* with its environment, i.e. making contact with and even manipulating, elements such as objects and other agents. In tasks that are both dynamic and interactive, RL offers a promising direction since human demonstrators can often struggle with precise, high-speed robot teleoperation, and hard-coded policies can be brittle when taken out of a controlled context.

We introduce a novel dynamic, interactive RL testbed that modifies air hockey, a popular game, with a collection of objects, Figure 1 illustrates some of the potential of this domain. By focusing on puck-hitting, the domain is inherently interactive and dynamic. This platform offers several advantages that facilitate RL training. The puck's constrained movement allows for efficient environment resets, while a strictly controlled agent workspace ensures safe autonomous operations when exploring. By incorporating multiple objects, both virtual and real, we can describe a wide array of tasks, illustrated in Figure 3. To evaluate RL without the physical setup we provide *two* simulators, illustrated in Figure 1, of increasing fidelity to the real world and tunable parameters so that sim-to-real transfer algorithms can be assessed even without the physical system. Finally, we introduce a real-world human-teleoperated dataset using two teleoperation systems for the agent, allowing the assessment of learning from demonstration and offline RL algorithms (see Figure 2).

This work not only describes the domain and associated tasks but also assesses several baseline algorithms and elucidates the key design decisions motivating the algorithm. We also assess behavior cloning, vanilla RL, and offline RL on several tasks in simulation and the real world. We demonstrate empirically that the set of tasks developed has a smooth variation from easy to difficult for all of these algorithms, both in simulation and the real world.

Researchers can use a wide range of tasks in a curriculum to test the capability of their algorithms, with simpler tasks in the 2D simulator to complex ones in Robosuite. This testbed offers an assessment of dynamic components and sparse interactions, properties that might not be assessed in other RL benchmarks. Furthermore, the testbed offers a tool for assessing a wide range of RL settings such as goal-conditioning, model learning, task transfer, skill learning, offline, and inverse reinforcement learning—to name just a few.

* denotes equal contribution, corresponding author calebc@cs.utexas.edu



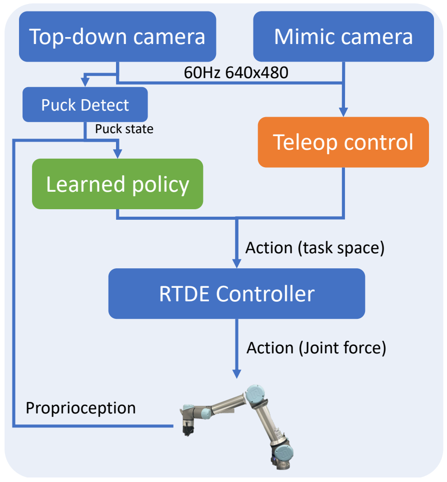


Fig. 1: (a) Robot Air Hockey is a testbed that contains dynamic, interactive Air Hockey tasks from multiple domains ranging from simulation to the real world. It is suitable for evaluating a variety of frameworks such as RL, learning from demonstrations, and skill transfer. (b) Overview of our control pipeline. We use an RTDE controller to transform the task actions into joint forces for the robot.

Finally, we intend to offer a real-world air hockey platform to assess high-performing algorithms in the real world, perform user studies, and even remote data collection.

II. Domain Description

A. Components

The common components across environments consist of the table, the paddle, the pucks, moveable objects such as blocks, and immovable obstacles. A robot arm is used in the Robosuite and real-world environments to control the paddle. In the 2D simulator the paddle is manipulated directly. In all domains, the table workspace is 66 in $\times$24 in, the paddle is 3.75in in diameter and the puck is 2.5 in. We maintain consistency between the domains for transfer learning.

B. Teleoperation

The mouse teleoperation setup streams a live video feed of the robot as seen by the overhead camera after performing an orthographic tomography to transform the camera coordinate system into the robot coordinate system in 2D. The mouse x, y position in the image is mapped to a desired robot x, y pose, using force control to maintain contact with the table.

Similarly, the shadowing setup tracks paddle movement and maps it to desired robot x, y positions. Instead of the mouse, the human manipulates a red paddle, and color segmentation is to track the paddle movement on this surface and transform these movements. This approach allows the user to play more naturally, as it is in direct control of a paddle, while bounded by the capabilities of the robot.

C. Environments

A common high-level reward interface and action space are used for all environments sim and real.

1) 2D: Our 2D simulation environment uses the Python implementation of Box2D [17] as a physics simulation back-end. While we are not able to deploy robot arm models in this environment, we use hand-tuned reward shaping so that sequences of actions are empirically more realistic for a robot arm. This environment has many changeable world parameters such as paddle mass, puck mass, dampening, friction, gravity, starting puck velocities, and many more parameters.

2) 3D: Our 3D simulation environment is a custom Robosuite [18] setting which builds on top of a MuJoCo [24] as a simulation backend and allows us to simulate the robot arm in addition to the other components such as the table, a paddle, and a puck. For control, the operational space controller [25] is available in Robosuite, which we modify to maintain stable contact with the table.

3) Real World: Our real-world setup consists of a Wind Chill air hockey table tilted 5.5 degrees and a UR5e robot arm. The vision system uses a Sony Playstation Eye, a high framerate camera, which gathers 640×480 frames at 60 FPS, mounted to the ceiling to have a full view of the table. The robot control operates at 20 FPS, which is dictated by the cost of vision processing and computing the desired action from the model while maintaining a stable control loop. We use hue-saturation-value (HSV) segmentation and a homography transformation using OpenCV [26] to estimate puck location and velocity. We visualize puck detection in Figure 10. We use force and operation space control on the robot to maintain the contact between the paddle and the table. A picture of our workspace and tasks is described in Fig 3. The control flow is illustrated in Figure 1b.

D. Tasks

We provide a collection of eleven total tasks with tuned rewards which vary in difficulty from those that can be achieved through behavior cloning (reaching) to those where even humans can struggle in the real world (juggling). Each task includes a designed reward function, and some can include additional objects beyond the puck and the end effector, such as target regions or other objects to be manipulated through the puck. We have five tasks on the real robot, six in Robosuite, and eleven in Box2D. The wide variety of tasks allows us

Environment	Method	Robot Air Hockey Tasks									
		Reach	Reach V.	Touch	Strike	Strike Crowd	Juggle	Puck V.	Block	Hit Goal	Hit Goal V.
Box2D [17]	BC	0.9	0.8	1.0	0.7	0.3	0.3	0.7	0.0	0.1	0.0
	IQL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.4	0.0
	RL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.9	0.0
Robosuite [18]	BC	0.9	0.8	0.8	-	-	0.6	0.6	-	0.1	-
	IQL	0.9	0.9	0.8	-	-	0.7	0.8	-	0.1	-
	RL	1.0	1.0	1.0	-	-	0.9	0.9	-	0.2	-
Real World	BC	0.9	0.1	0.3	-	-	-	0.1	-	-	-
	IQL	1.0	0.0	0.6	-	-	-	0.3	-	-	-
	Human	1.0	0.0	1.0	-	-	0.3	1.0	-	-	-

TABLE I: Success rates for the Box2D, robosuite and real robot environment. A dash (-) indicates that this combination of environment, method and task was not evaluated. For Box2D, a combination of SAC/HER [19], [20] is used to train the goal-conditioned RL policies, while PPO [21] is used to train RL policies for the remaining tasks. For Robosuite, we use PPO for Vanilla RL, following the implementation of CleanRL library [22]. In simulation, BC and IQL [23] are used to learn policies offline using the ”expert” data collected with trained PPO policies. On the real robot, we instead use a teleoperated dataset collected by 8 human players of varying skill of 400 trajectories for BC and IQL.

not only to assess varying difficulty but also leaves room for potential skill learning and transfer learning. The description of the tasks and their rewards are in Appendix V-B.

E. *Offline Data*

On the real robot, we provide a dataset of human gameplay using both teleoperation methods (350 mouse-teleop trajectories and 50 shadow-teleop trajectories) gathered from 8 participants of varying skill. We visualize some of the human-gathered demonstrations in Figure 5 and Figure 10. In these, it is clear that the human demonstrators, while able to hit the puck at least once, can struggle to achieve multiple hits. In Table I, we assess successful juggling as hitting the puck at least four times in a trajectory, and we can see that humans only achieve moderate success. We discuss real-world data collection and visualize human-gathered policies in Figure 5 and Appendix V-J.

III. Learning Methods

In our experiments, we evaluate three representative methods: Behavior cloning (mean squared error loss), Vanilla Reinforcement Learning (soft actor-critic and proximal policy algorithms), and Offline Reinforcement Learning (IQL). Behavior cloning learns a policy to take desired actions, Reinforcement learning maximizes discounted reward, and offline RL assumes environment interactions are not available to the agent. An extended discussion can be found in Appendix V-C.

IV. Experimental Results

In this section, we briefly describe the qualitative results of running various learning algorithms in the three domains. For all experiments, our policies are represented with multi-layer perceptions with a hidden layer of 256. We include several figures illustrating the robot behavior in Box2D (Figure 4), Robosuite (Figure 6) and the real world (Figure 5). We also discuss training curves in Appendix V-I.

A. *Vanilla Reinforcement Learning*

We assess Vanilla RL in simulation providing sufficient reward shaping to ensure that the agents can perform well, and find good performance in both simulators, as seen in Table I. In this table, we also describe the specific RL algorithms we employ. This provides evidence to validate the hypothesis that RL is an ideal choice in dynamic, interactive tasks where behavior cloning might struggle to utilize the nuanced distinctions in capabilities. In the real world, however, the random exploration necessary for RL from scratch prevents the robot from gathering enough meaningful feedback without wearing down the robot.

B. *Behavior Cloning*

In the real world, using the dataset of approximately 128,000 frames of human-generated striking behavior gathered through two teleoperation systems, we train a network that maps puck and proprioceptive state information to actions. Because humans can struggle to hit using teleoperation, the learned models likewise struggle, while also being hampered by distribution mismatch. In the real world, we also experimented with an image-based policy using a ResNet-18. In the simulated domains, we behavior clone using 1M time steps from high-performing policies.

C. *Offline RL*

Offline RL offers a mechanism for utilizing data collected offline from the agent, such as our teleoperation dataset, to learn a policy. For offline RL, we use IQL [23]. Because Offline RL does not generate its own environment interactions, though, it can struggle more with distribution mismatch. Nonetheless, Offline RL generally outperforms Behavior Cloning in the real world.

V. Conclusion

We introduce an air-hockey-based evaluation domain for RL to evaluate RL in dynamic interactive environments. We provide the initial set of evaluations that support the hypothesis that in these kinds of domains, RL can outperform imitation learning in a real-world robotics setting. Most importantly, we intend for researchers to use the suite of tasks, simulators, and collected human data for assessing a wide variety of RL settings from goal-conditioned RL to unsupervised skill learning to sim-to-real transfer, which we explore more in Appendix V-H. Videos describing the project, of learned policies and human data can be found at RLAirHockey.github.io.

APPENDIX

A. Task Figures

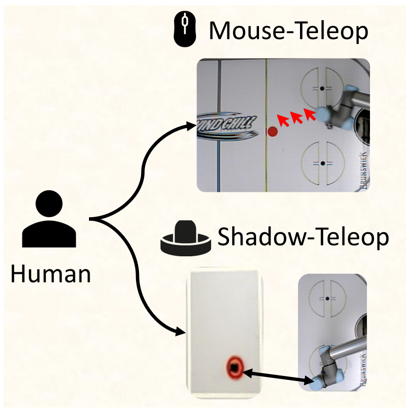


Fig. 2: Robot Air Hockey supports two types of teleoperations. Mouse-Teleop (top): The user moves the mouse to control the robot. Shadow-Teleop (bottom): the user moves a paddle to control the robot.

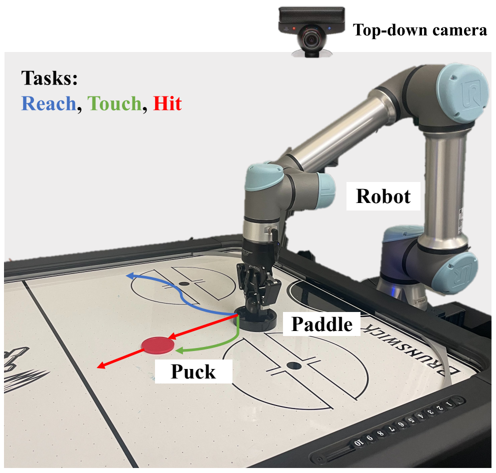


Fig. 3: Robot Air Hockey real-world setup. We use a top-down camera to provide observation and a UR5e robot to actuate the paddle. Our real-world setup can facilitate many air hockey tasks, including but not limited to reaching, touching, and hitting.

B. Task Descriptions

Below we describe the ten tasks we assessed in this work. In addition to the reward specified by the task itself, we also provided regularization to ensure that undesirable behaviors such as jittering and twisting motion, which can cause the robot to emergency-stop, were prevented.

Below we describe the eleven tasks we assessed in this work. In addition to the reward specified by the task itself, we also provided regularization to ensure that undesirable behaviors such as jittering and twisting motion, which can cause the robot to emergency stop (a safety measure on the UR5 that prevents it from damaging itself or the human), were prevented.

1) **Reaching**: The paddle reaches a random location. When the paddle is within ϵ of the goal, the agent receives a positive reward and the episode is reset.
2) **Reaching with Velocity**: The paddle reaches a random location *and velocity* pair. When the paddle is within $\epsilon_{\text{position}}, \epsilon_{\text{velocity}}$ of the goal (both position and velocity) a reward is given, and the episode is reset.
3) **Touching**: The paddle touches the puck. Upon detection of contact, a reward is given. The agent is continually rewarded each time it touches the puck.
4) **Striking a stationary puck**: The paddle hits a stationary puck and moves it a minimum distance. If a sufficient velocity is achieved, a reward is given and the episode ends.
5) **Striking a stationary puck into a crowd**: The paddle hits a stationary puck which causes it to collide with blocks, similar to the game of pool. The reward is dependent on the amount of spread from the blocks from the crowd.
6) **Juggling**: The paddle hits a puck a minimum distance above the paddle at the time of hitting.
7) **Puck Velocity**: The paddle hits a puck and causes it to move at a minimum upward velocity.
8) **Moving a block**: The paddles hits a puck into a block, causing it to move a minimum distance from the block's initial position.
9) **Hitting into a goal region**: The paddle hits a puck into a goal circle region with a constant radius.
10) **Hitting into a goal region with desired velocity**: The paddle hits a puck into a goal circle region with a constant radius with a specified velocity. It is rewarded based on distance to the goal region's center, cosine similarity between the puck's vector when entering the goal region, and difference in the puck's magnitude from the desired velocity's magnitude.

While not a task we trained models for in this work, our Box2D simulation environment also supports both collaborative play and adversarial play, extending these environments as a potential testbed for multi-agent RL. Furthermore, additional tasks with increasing complexity can be easily constructed.

C. Learning Methods

In our experiments, we evaluate three representative methods: Behavior cloning (mean squared error loss), Vanilla Reinforcement Learning (soft actor-critic and proximal policy algorithms), and Offline Reinforcement Learning (IQL). Behavior cloning learns a policy to take desired actions, Reinforcement learning maximizes discounted reward, and offline RL assumes environment interactions are not available to the agent.

D. Preliminaries

A Markov decision process is defined by the tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, p, R)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space and $s \in \mathcal{S}, a \in \mathcal{A}$ are states and actions respectively. $p(s'|s,a)$ is the transition function that gives the probability of the next state s' given the current state and action (s,a). The reward function $R(s,a)$ maps state and action to a scalar reward. A policy $\pi(a|s)$ is the probability of an action given the current state.

E. Behavior Cloning

Behavior cloning [27] is an offline imitation learning approach that learns a policy using supervised learning. We use the common MSE loss to learn π:

$$\mathcal{L}_{\text{bc}} = \mathbb{E}_{(a,s)\sim\mathcal{D}}[||\pi(s) - a||_2^2].$$

Although the performance of behavior cloning is limited by the quality of the dataset and is plagued by compounding error problems in the limited data regime, it presents a simple and scalable alternative [12], [28] to the more complicated yet unstable imitation learning methods [14], [29], [30].

F. Reinforcement Learning

The goal of Reinforcement Learning is to learn a policy maximizing returns obtained on any user-specified reward function. Formally, RL aims to learn a policy π that maximizes the cumulative return $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma_t r(s_t, a_t))]$. RL reduces the effort of designing hand-tuned controllers by proposing general approaches that learn any behaviors.

Online RL: In our work, we rely on Proximal Policy Optimization (PPO) [21] as our baseline algorithm. PPO has been extensively used in robotics for its stability and guarantee of near-monotonic improvement.

Offline RL: Offline RL deals with a specific setting where environment interactions are not available to the agent, but rather the agent is provided with data of offline transitions of the form {s,a,r,s'} from the environment. This setting reduces the burden of exploration considerably and provides a safe way to learn from previously collected datasets. Offline deep RL algorithms suffer optimization difficulties due to problems like overestimation [31], and feature-coadaptation [32] leading to popular regularizers like pessimism [15], [33], [34]. We use a representative and performant algorithm for offline RL in this work - IQL [23].

IQL is a modification on the standard actor-critic learning procedure to replace the maximization of the state-action value function with a maximization that only chooses values for actions seen in the dataset.

G. Related Works

The field of agile robotics has seen growing interest in the past several years. Air hockey-playing robots have been studied in several settings [35]–[45]. Notably, a recent workshop [46] highlighted robot air hockey as a competitive robot learning task with several reports on model-based and deep learning methods [47]–[49]. Specifically, [35] used Bayesian optimization and a slew of other well-tuned solvers to plan trajectories for striking the puck. This work is distinct from these systems in the focus on a *variety* of tasks rather than highlighting competitive air hockey, as well as the incorporation of human teleoperation modes to assess demonstration and offline methods. The many works on robotic air hockey cover a wide range of topics and are summarized here:

1) [38] uses a hierarchical switching controller to play with different behaviors according to the robot's opponent.
2) [39] aims to generate motion plans that take advantage of weak spots in a human's vision when switching gaze.
3) [40] is an older study demonstrating the difficulty of building a vision-based autonomous air-hockey system.
4) [41] presents a state-model-based method to create air hockey-playing agents against which humans can compete.
5) [42] introduces a method to estimate the parameters of air hockey tables (e.g. coefficient of friction, etc.) automatically so that a robot may adapt its behavior accordingly.
6) [43] demonstrates a fast-hitting, air hockey-playing robot using analytical controllers that were developed by modeling the air hockey environment.
7) [44] design a high-speed wrist mechanism as an add-on to a manipulator so that a robot can easily compete with the speed of a human air hockey player.
8) [45] introduces a method to align simulators with real-world observations using causal relationships

Further, robotic table tennis [50], [51] has shown promise as a challenging domain for using RL to learn agile robot policies. The authors in [50] demonstrate that a combination of learning techniques, including evolutionary strategies, is important to learn high-performing table tennis policies. While our testbed is similar to that in [50] in terms of necessary agility, our air hockey is distinguished by the tilted table which induces a constant motion of the puck, regardless of the robot's intervention. This facilitates a wide range of tasks and allows automatic resets when learning since the puck continually returns to the reachable workspace of the robot.

H. Future Work And Relation with Manipulation Skills

In this section, we discuss the wide range of possible RL settings where the testbed can be used. Before jumping into these settings, we discuss the way in which a researcher might use this testbed. By providing tasks in the 2D simulator that can be run quickly and should be easy to solve, such as reaching or reaching with velocity, a researcher can sanity check their implementation. Then, based on the particular characteristics of their algorithm, such as goal-conditioning, object generalization, or transfer properties etc., they can take a collection of the tasks in the 2D domain and use them to assess their algorithm in a simplified environment that assesses the characteristic behaviors of the method. To illustrate scaling, they can then take those algorithms and apply them to the 3D simulator. Since the simulator shares the same state and action space as the real robot, if their method is offline then it can be trained with the offline real robot data, and possibly tested on the physical system.

Perhaps the most obvious setting for RL in air hockey is **goal-conditioned RL**. The inverted table is inherently goal-conditioned in the sense of striking the puck to a desired position, and one of the most challenging of our existing tasks is striking the puck to a desired position *with* a desired velocity. However, many more goal-conditioned settings exist, including using the puck to hit an object to a location, achieving a sequence of goals for the end effector, or getting the puck into a desired goal state relative to other objects.

Another clear setting for future RL assessment of the robot air hockey testbed, considering the multiple simulation environments of increasing fidelity, is to utilize offline data generated by a higher fidelity simulator to train policies using another simulator [52]. **Sim-to-real transfer** can be iterated on quickly through 2D-to-3D sim-to-sim transfer and then tested on 3D-to-real transfer. Similarly, **model-based RL** can utilize the simpler simulated environment models and transfer to model more complex real-world dynamics, especially using the offline data.

Another consequence of the paired simulators is to provide a natural curriculum for **curriculum learning**. Within single tasks, depending on the initialization of target objects tasks can range from easy to hard, as well as between simulators. Also, because many of the tasks are derivative of each other, this leaves the room open for **transfer learning or meta-learning**, where learning on some tasks will benefit downstream learning. As an extension, **unsupervised learning or skill learning** make a lot of sense in many of the environments since certain strikes and hits will be useful across multiple tasks. While hard-coding these behaviors would be challenging, learning them from the offline dataset or from experience could allow agents to perform complex behaviors like hitting a target block to a desired location.

The air hockey setting itself offers possibilities for object-based factorization. While this work only provided a few additional objects beyond the puck and paddle, a wider variety of objects are actively being incorporated to investigate **object-based generalization** algorithms. For example, adding in target objects to push, obstacles for the puck, negative or positive reward regions, or even regions of different physics are all possible. Furthermore, by modifying real-world parameters such as table angle or paddle shape, we can assess causal or model-based RL. Finally, especially as more objects are added, visual complexity opens the door for more complex object detection than simple color segmentation.

With many manipulation tasks focusing on tasks that can often be performed with high precision for human demonstrators, such as pick-and-place, this system offers the opportunity for **superhuman performance**, especially on even more challenging tasks, such as two-puck juggling. RL is an ideal tool for this setting, as suggested by the results in this work, but RL directly on the robot is limited.

While the existing system shows a wide range of capabilities, the real-world system illuminates several insights about running learned policies in a **high-speed setting**. In particular, maintaining high latency is challenging, and we found that even humans playing at 20Hz can struggle. We hope to follow up with experiments investigating human capabilities in teleoperation. Furthermore, one of the most significant hurdles is the robot emergency-stopping, either because a learned policy takes cyclic actions that result in damaging resonant behavior, or because the robot jerks too quickly when changing direction. Further investigation into action smoothing could provide higher-quality human demonstration data and offline policies.

As an immediate direction, the air hockey testbed offers **offline RL** assessment in a dynamic, interactive real-world set of tasks. The initial results suggest that RL is preferable because of the difficulty of gathering high-quality demonstrations, and further investigation into offline and mixed offline and online methods can provide substantial benefits.

We are looking into **multi-agent settings**. Air hockey is itself an adversarial game. Beyond this, simulators can be modified to include collaborative settings such as rallying the puck between multiple paddles or multiple puck juggling with multiple paddles. Furthermore, we can extend the adversarial setting to more than two goal regions and opposing paddles in a free for all. On the physical robot, by incorporating an additional UR5 at the opposite end of the table, we can realize some of these settings. Alternatively, while humans far outstrip the physical capabilities of the UR5, adversarial play can be between two humans playing against each other through the robot.

Finally, the teleoperation systems offer several means of assessing **human-robot interaction**. Some possible directions for this include simple assessments of human skill levels over play on the robot or frustrations with robot capabilities, to more complex assessments of humans being able to detect or react to another human playing, or a learned policy.

In summation, we believe that the robot air hockey testbed offers ample opportunities to assess a significant range of RL capabilities across multiple domains. While there are certainly limitations, a shared interface can facilitate proof of concept work.

I. Training Curves

We illustrate the training curves as normalized reward over the number of timesteps for Box2D (Figure 7), success rate over 15 episodes over the number of training timesteps for Robosuite (Figure 8), mean squared error for behavior cloning on the real robot, and actor loss for IQL over number of iterations of training (Figure 9). The real robot uses different losses because we do not evaluate during training, as that would require loading partial networks onto the real robot, which would greatly extend training time.

J. Puck Hitting dataset

This section will describe how we extract the state of the puck on the table from the collected videos and show exemplary trajectories from our dataset. We collected videos of several participants attempting to hit the puck after being dropped from the far end of the table using the various teleoperation modalities.

Puck state extraction. We use red or green pucks because they are the easiest to locate on the air hockey table using HSV color segmentation. To find the location of the puck on the air hockey table, we first apply a homography transformation to the image to account for the distortions introduced by the camera. We then apply a mask to each frame that indicates whether a pixel falls within the color bounds for the red or green puck. Lastly, we find the median pixel location of all the masked pixels and apply an affine transform that maps from the pixel location in the image to the robot base's reference frame. **Puck trajectories** In Figure 10, we illustrate the extracted trajectories from shadow-teleop and mouse-teleop. Gaps in the puck's trajectory indicate where the puck was not detected, for example, if it was occluded by the robot.

References

[1] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[3] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo *et al.*, "Octo: An open-source generalist robot policy," 2023.

[4] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. Julian, C. Finn *et al.*, "Actionable models: Unsupervised offline reinforcement learning of robotic skills," *arXiv preprint arXiv:2104.07749*, 2021.

[5] H. Sikchi, R. Chitnis, A. Touati, A. Geramifard, A. Zhang, and S. Niekum, "Score models for offline goal-conditioned reinforcement learning," *arXiv preprint arXiv:2311.02013*, 2023.

[6] B. Eysenbach, S. Levine, and R. R. Salakhutdinov, "Replacing rewards with examples: Example-based policy search via recursive classification," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11 541–11 552, 2021.

[7] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," *arXiv preprint arXiv:1802.06070*, 2018.

[8] S. Park, O. Rybkin, and S. Levine, "Metra: Scalable unsupervised rl with metric-aware abstraction," *arXiv preprint arXiv:2310.08887*, 2023.

[9] T. Zahavy, Y. Schroecker, F. Behbahani, K. Baumli, S. Flennerhag, S. Hou, and S. Singh, "Discovering policies with domino: Diversity optimization maintaining near optimality," *arXiv preprint arXiv:2205.13521*, 2022.

[10] C. Chuck, K. Black, A. Arjun, Y. Zhu, and S. Niekum, "Granger-causal hierarchical skill discovery," *arXiv preprint arXiv:2306.09509*, 2023.

[11] C. Chuck, S. Chockchowwat, and S. Niekum, "Hypothesis-driven skill discovery for hierarchical deep reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5572–5579.

[12] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[13] J. Hejna, R. Rafailov, H. Sikchi, C. Finn, S. Niekum, W. B. Knox, and D. Sadigh, "Contrastive prefence learning: Learning from human feedback without rl," *arXiv preprint arXiv:2310.13639*, 2023.

[14] H. Sikchi, A. Saran, W. Goo, and S. Niekum, "A ranking game for imitation learning," *arXiv preprint arXiv:2202.03481*, 2022.

[15] H. Sikchi, Q. Zheng, A. Zhang, and S. Niekum, "Dual rl: Unification and new methods for reinforcement and imitation learning," *arXiv preprint arXiv:2302.08560*, 2023.

[16] T. Ni, H. Sikchi, Y. Wang, T. Gupta, L. Lee, and B. Eysenbach, "f-irl: Inverse reinforcement learning via state marginal matching," in *Conference on Robot Learning*. PMLR, 2021, pp. 529–551.

[17] E. Catto. [Online]. Available: https://box2d.org/

[18] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu, "robosuite: A modular simulation framework and benchmark for robot learning," *arXiv preprint arXiv:2009.12293*, 2020.

[19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[20] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[22] S. Huang, R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta, and J. G. Araújo, "Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms," *Journal of Machine Learning Research*, vol. 23, no. 274, pp. 1–18, 2022. [Online]. Available: http://jmlr.org/papers/v23/21-1342.html

[23] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv*, 2021.

[24] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[25] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[26] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[27] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural computation*, vol. 3, no. 1, pp. 88–97, 1991.

[28] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.

[29] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[30] S. K. S. Ghasemipour, R. Zemel, and S. Gu, "A divergence minimization perspective on imitation learning methods," in *Conference on robot learning*. PMLR, 2020, pp. 1259–1277.

[31] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration. corr abs/1812.02900 (2018)," *arXiv preprint arXiv:1812.02900*, 2018.

[32] A. Kumar, R. Agarwal, T. Ma, A. Courville, G. Tucker, and S. Levine, "Dr3: Value-based deep reinforcement learning requires explicit regularization," *arXiv preprint arXiv:2112.04716*, 2021.

[33] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[34] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[35] P. Liu, D. Tateo, H. Bou-Ammar, and J. Peters, "Efficient and reactive planning for high speed robot air hockey," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 586–593.

[36] A. Taitler and N. Shimkin, "Learning control for air hockey striking using deep reinforcement learning," 2017.

[37] J. Jankowski, A. Marić, and S. Calinon, "Airlihockey: Highly reactive contact control and stochastic optimal shooting," 2024.

[38] A. Namiki, S. Matsushita, T. Ozeki, and K. Nonami, "Hierarchical processing architecture for an air-hockey robot system," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1187–1192.

[39] M. Ogawa, S. Shimizu, T. Kadogawa, T. Hashizume, S. Kudoh, T. Suehiro, Y. Sato, and K. Ikeuchi, "Development of air hockey robot improving with the human players," in *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, 2011, pp. 3364–3369.

[40] B. Bishop and M. Spong, "Vision-based control of an air hockey playing robot," *IEEE Control Systems Magazine*, vol. 19, no. 3, pp. 23–32, 1999.

[41] A. AlAttar, L. Rouillard, and P. Kormushev, "Autonomous air-hockey playing cobot using optimal control and vision-based bayesian tracking," in *Towards Autonomous Robotic Systems*, K. Althoefer, J. Konstantinova, and K. Zhang, Eds. Cham: Springer International Publishing, 2019, pp. 358–369.

[42] H. Alizadeh, H. Moradi, and M. N. Ahmadabadi, “Automatic calibration of an air hockey robot,” in *2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, 2013, pp. 107–112.

[43] P. Liu, D. Tateo, H. Bou-Ammar, and J. Peters, “Efficient and reactive planning for high speed robot air hockey,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 586–593.

[44] K. Tadokoro, S. Fukuda, and A. Namiki, “Development of air hockey robot with high-speed vision and high-speed wrist,” *Journal of Robotics and Mechatronics*, vol. 34, no. 5, pp. 956–964, 2022.

[45] P. Huang, X. Zhang, Z. Cao, S. Liu, M. Xu, W. Ding, J. Francis, B. Chen, and D. Zhao, “What went wrong? closing the sim-to-real gap via differentiable causal discovery,” in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 06–09 Nov 2023, pp. 734–760. [Online]. Available: https://proceedings.mlr.press/v229/huang23c.html

[46] “Robot air hockey challenge 2023,” https://air-hockey-challenge.robot-learning.net/home, accessed: 2024-03-28.

[47] A. Orsula, “Learning to play air hockey with model-based deep reinforcement learning,” *Air Hockey Challenge at Advances in neural information processing systems*, 2023.

[48] M. E. B. V. de Bakker3Atalay, D. Ö. E. Yagmurlu, M. F. Z. J. D. Yang, H. Zhou, X. Jia, O. Celik, F. Otto, R. Lioutikov, and G. Neumann, “Air hockey challenge 2023: Air-hockit team report,” *Air Hockey Challenge at Advances in neural information processing systems*, 2023.

[49] F. Minnucci, “Applying rule-based controllers and reinforcement learning to control a general purpose robot: the air hockey challenge case,” *Air Hockey Challenge at Advances in neural information processing systems*, 2023.

[50] T. Ding, L. Graesser, S. Abeyruwan, D. B. D’Ambrosio, A. Shankar, P. Sermanet, P. R. Sanketi, and C. Lynch, “Learning high speed precision table tennis on a physical robot,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 10 780–10 787.

[51] D. D’Ambrosio, N. Jaitly, V. Sindhwani, K. Oslund, P. Xu, N. Lazic, A. Shankar, T. Ding, J. Abelian, E. Coumans, G. Kouretas, T. Nguyen, J. Boyd, A. Iscen, R. Mahjourian, V. Vanhoucke, A. Bewley, Y. Kuang, M. Ahn, D. Jain, S. Kataoka, O. Cortes, P. Sermanet, C. Lynch, P. Sanketi, K. Choromanski, W. Gao, J. Kangaspunta, K. Reymann, G. Vesom, S. Moore, A. Singh, S. Abeyruwan, and L. Graesser, “Robotic table tennis: A case study into a high speed learning system,” in *Robotics: Science and Systems XIX*, ser. RSS2023. Robotics: Science and Systems Foundation, Jul. 2023. [Online]. Available: http://dx.doi.org/10.15607/RSS.2023.XIX.006

[52] J. P. Hanna, S. Desai, H. Karnan, G. Warnell, and P. Stone, “Grounded action transformation for sim-to-real reinforcement learning,” *Machine Learning*, vol. 110, no. 9, pp. 2469–2499, 2021.

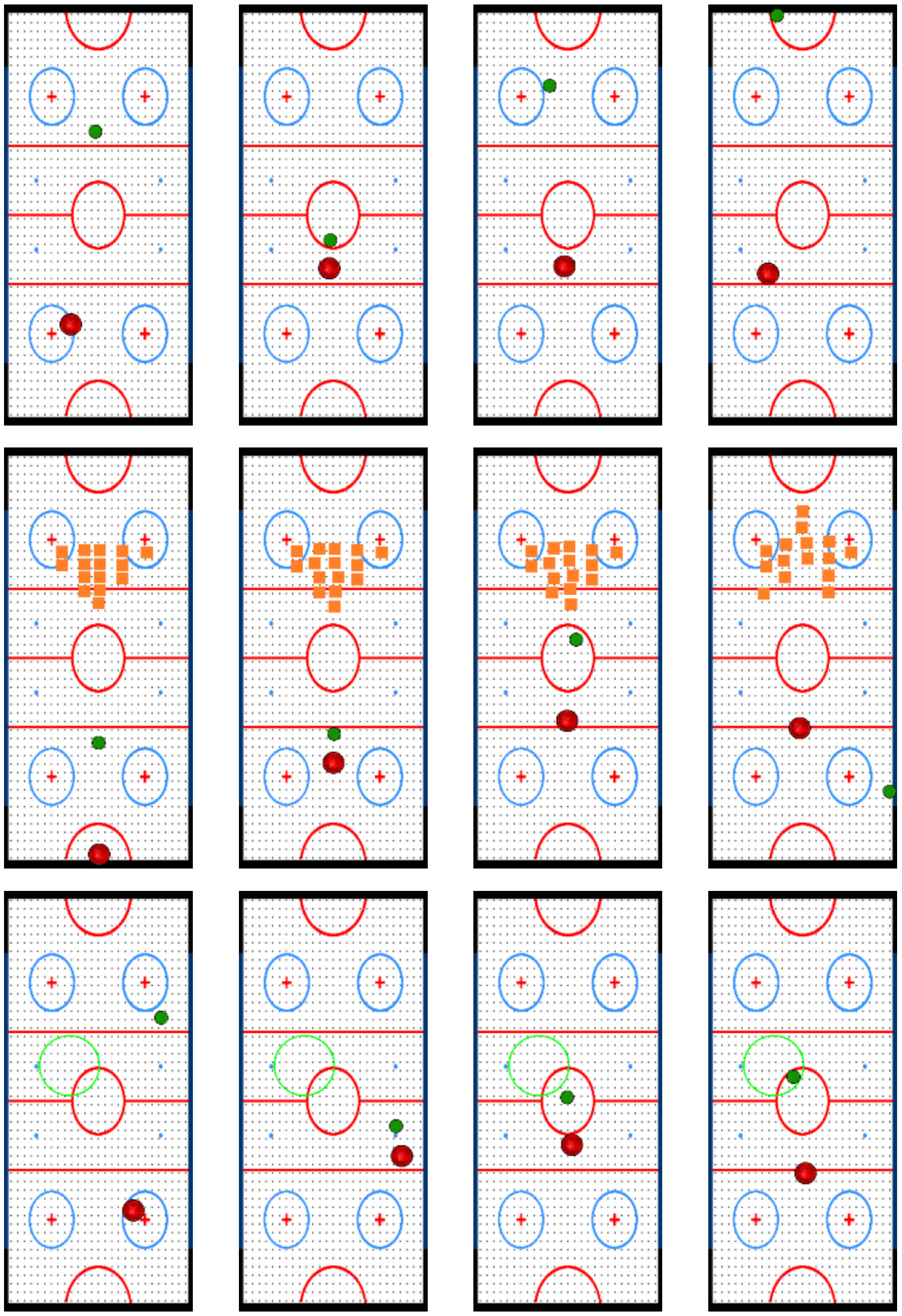


Fig. 4: Execution rollouts in Box2D for various tasks. For each first frame the motion of the puck is downwards. Top row: The task where the policy tries to hit the puck to reach a minimum amount of upward velocity. Middle row: The task where the policy hits the puck into a crowd of blocks, causes them to spread. Bottom row: The task where the policy moves a puck into a goal region, shown as a green circle.

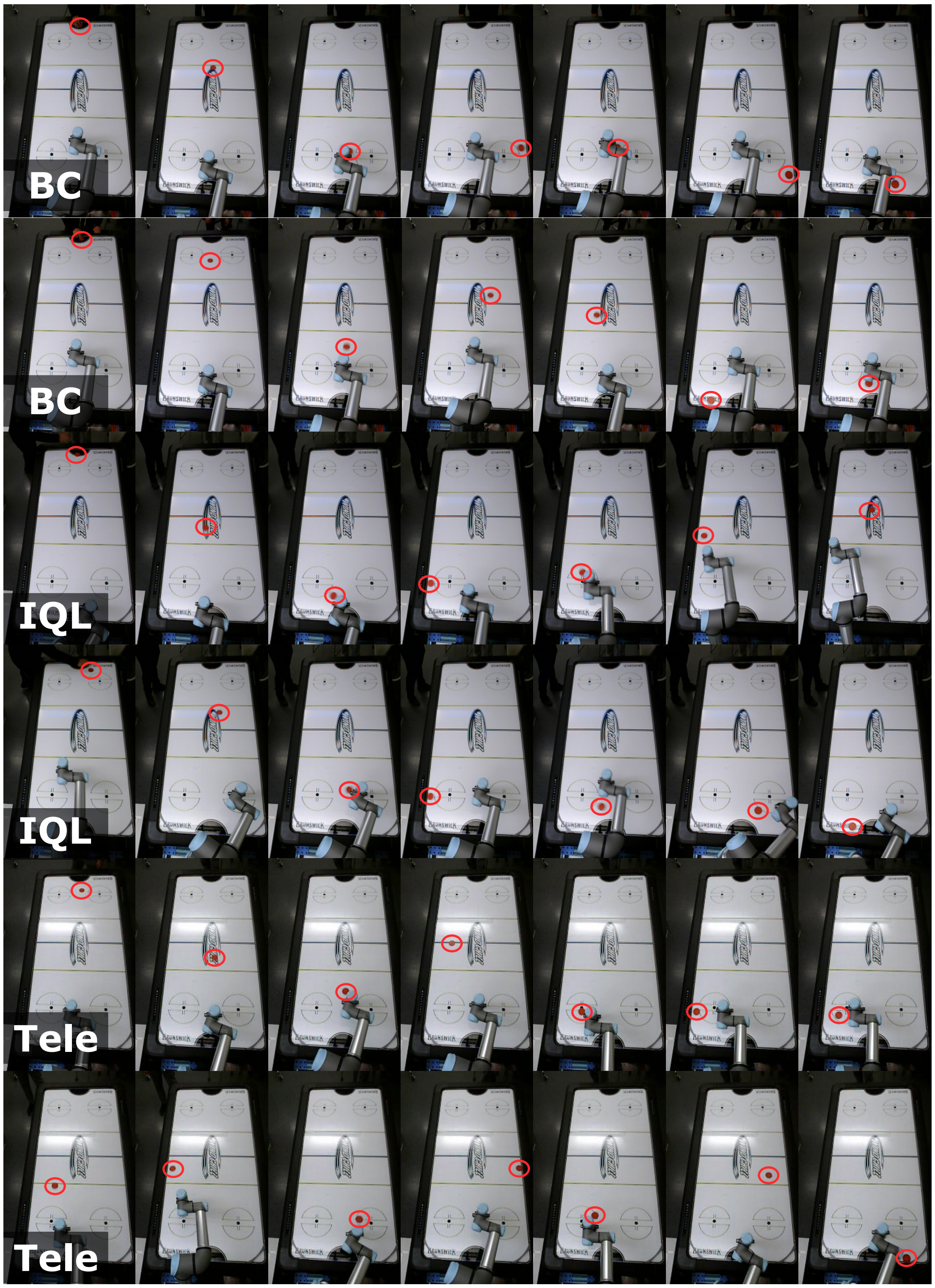


Fig. 5: Execution rollouts on the UR5 air hockey setup for policies trained with behavior cloning, IQL with the touching loss, and human teleoperation demonstrations. Notice that even though humans are trying to achieve multiple bounces, they often hit the puck too erratically to effectively return, so demonstrations can vary significantly in skill, even after cleaning human failure modes.

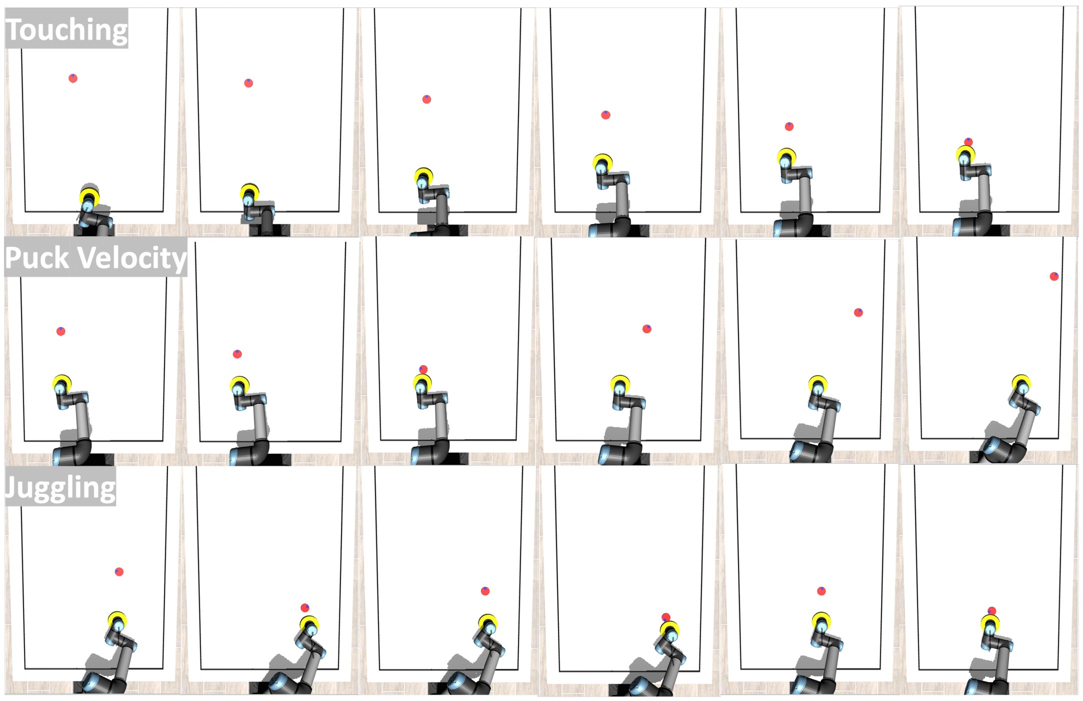


Fig. 6: Execution rollouts in Robosuite simulator. The puck is circled in red for emphasis. Policies are trying to touch the puck, hit the puck with a minimum upward velocity, and juggle the puck.

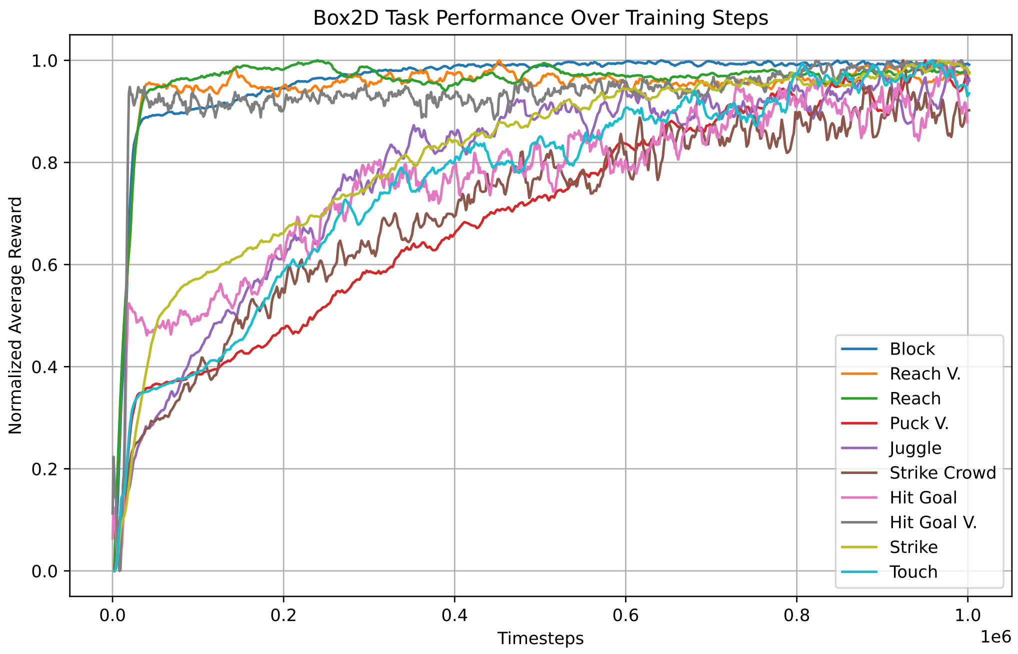


Fig. 7: Training curve for all tasks in Box2D. For vanilla RL tasks, the rewards are averaged across 5 seeds, while for goal-conditioned RL we use 1 seed. Rewards are then normalized to to the [0, 1] range with respect to the minimum and maximum reward seen for each task. Tasks in which performance converges quickly indicate that the task is either trivially easy (reaching a position with the paddle, reaching a position with the paddle with a desired velocity) or too difficult (moving a block, hitting a puck into a goal position with desired velocity).

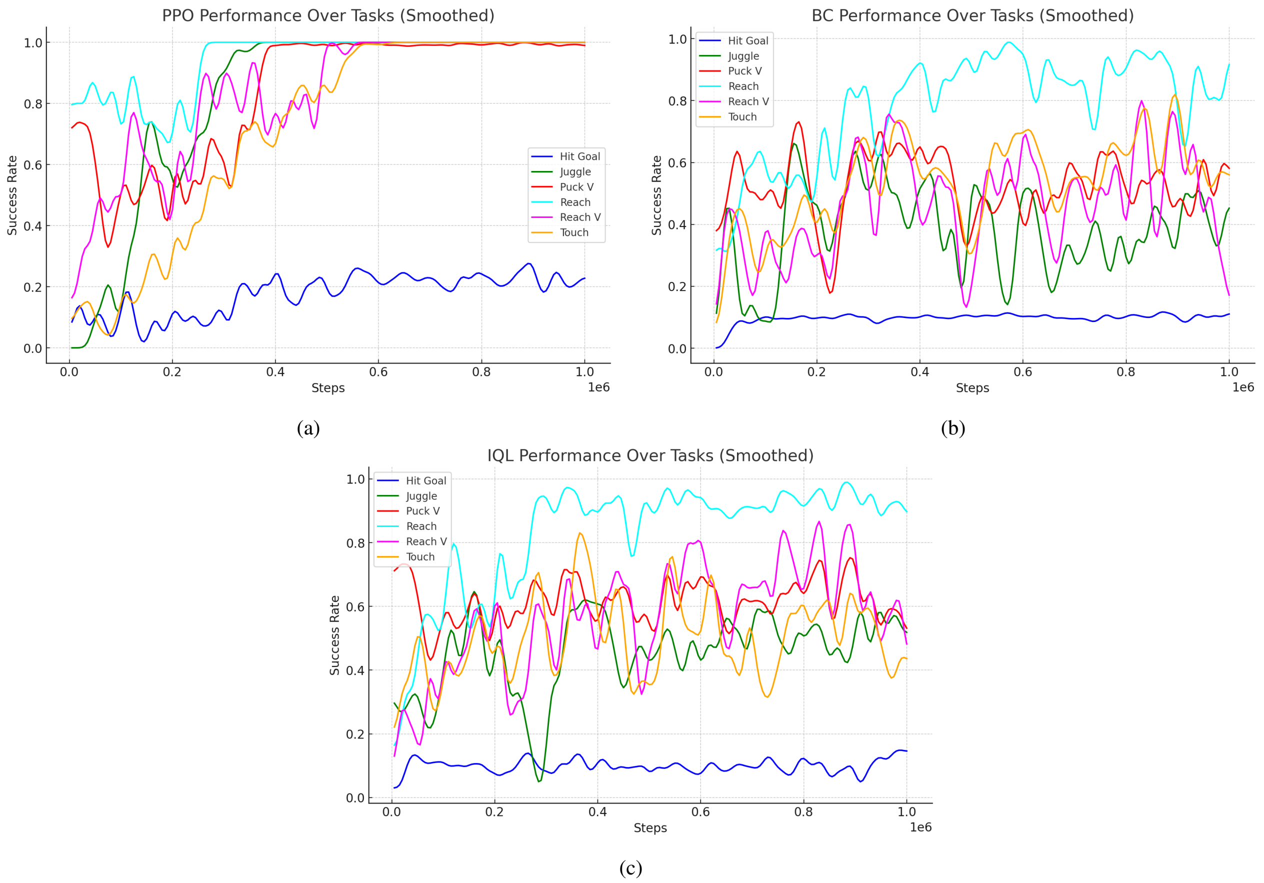


Fig. 8: Robosuite Training curves. **(a)**: Vanilla RL with PPO, following the implementation of CleanRL library [22]. **(b)** Behavior Cloning with data collected the "expert" policy (a trained PPO policy). **(c)** IQL [23] using the same offline data, using asymmetric τ set to 0.6.

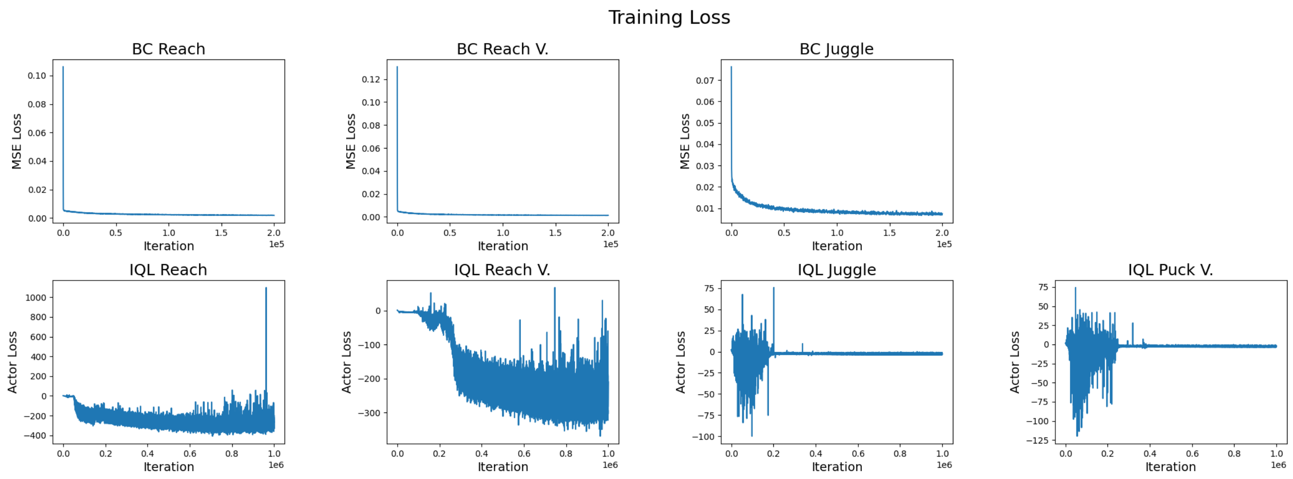
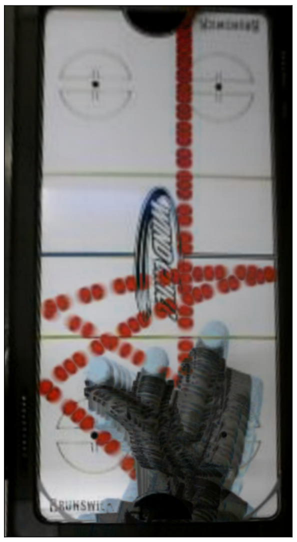
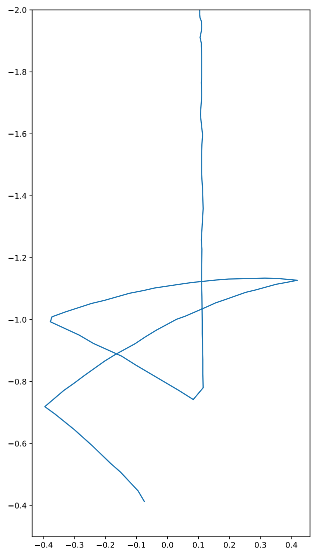
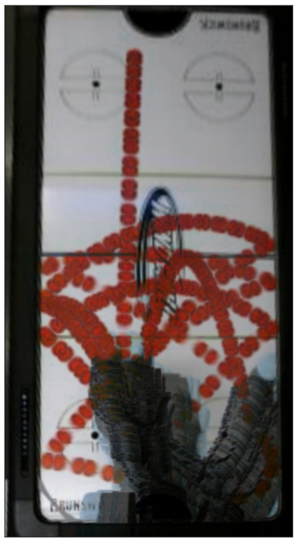


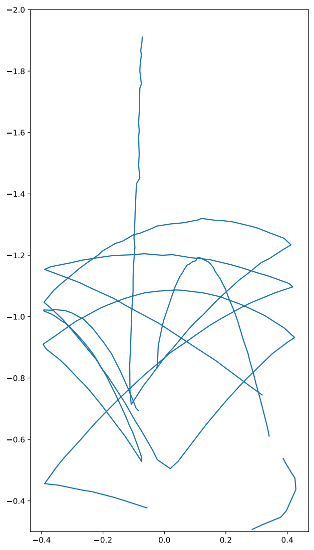
Fig. 9: Loss (MSE of behavior cloning and Actor Loss for IQL) curves for training using data collected from the real robot. Since humans did not distinguish puck velocity and juggling, we only trained a single behavior cloning policy (BC Juggle).


(a) Stacked images representing the video of a trajectory collected with **Shadow-teleop**



(b) Extracted state trajectory of puck data collected with **Shadow-teleop**


(c) Stacked images representing the video of a trajectory collected with **Mouse-teleop**



(d) Extracted state trajectory of puck data collected with **Mouse-teleop**

Fig. 10: Puck-hitting trajectories collected with teleoperation modalities. Generally, mouse-teleop is more responsive, and thus participants could strike the puck more easily. Nonetheless, participant skill was the primary factor when assessing the quality of a gathered trajectory.