HRI Gym: A Tool for Crowdsourcing Human-Robot User Studies

Anonymous authors

Abstract—**While the field of HRI is grounded in learning from humans, it is usually challenging to collect large amounts of user input. This makes it difficult to run, evaluate, and reproduce HRI studies. In response, we present an open-sourced online framework that enables HRI researchers to design experiments, collect large amounts of user data, have access to a library of HRI algorithms, and evaluate their studies. By creating a core API and building general components for running both interactive and non-interactive experiments, we enable researchers to export a wide variety of HRI studies to the web. Furthermore, by providing tools to visualize results and collect user evaluation, we simplify the process of collecting massive amounts of human feedback. To test this tool, we apply it to several imitation learning tasks. We conclude by discussing the challenges of performing HRI studies over the web and delineating future directions for the HRI Gym tool.**

Index Terms—**Crowdsourcing, Online Framework, Massive Demonstrations**

I. Introduction

In recent years, we have seen many advances in the field of human-robot interaction (HRI). Novel algorithms have opened up new and promising directions in understanding human-robot interaction, collaboration, intent inference, adaptation, and a variety of other problems [1]. However, the HRI field has often been criticized for developing algorithms without proper validation. This has led some to question the validity of HRI approaches in general.

We believe that both the development and the validation of HRI algorithms should be based on extensive user feedback. In order to have robots that can safely and meaningfully interact with humans, different interaction algorithms need to be compared and tested on large user populations. In addition, data from such a large user population should be applied to develop algorithms that can generalize across a variety of HRI tasks.

However, this presents a host of challenges. First, in order to carry out large physical studies, research labs have to invest a considerable amount of time, labor, and resources. In turn, as robotic studies get larger, it becomes more difficult to change experimental procedure and analysis, meaning that experimenters often do not have the chance to learn from user feedback. Second, a lack of standardization in algorithms, hardware, experimental setup, and evaluation procedure makes it difficult, if not impossible, to test and compare HRI methods. This has created a reproducibility crisis in the HRI community.

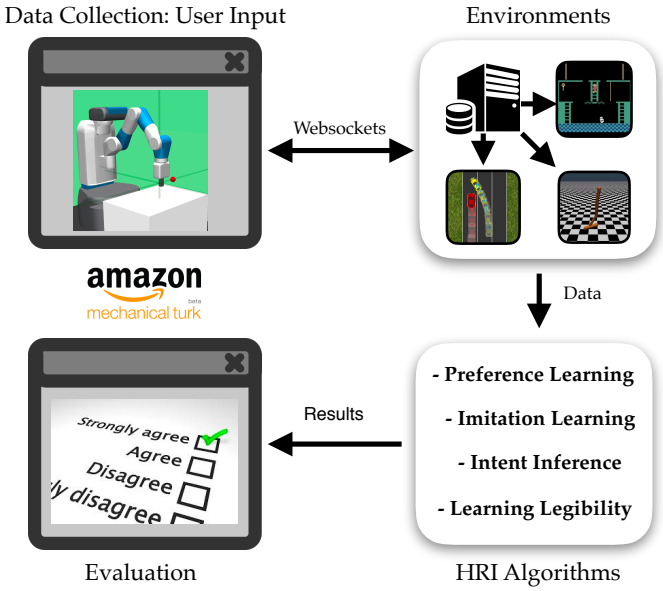


Fig. 1: A summary of how HRI Gym is structured around four abstract components of HRI research: user inputs, environments, HRI algorithms, and evaluation.

To alleviate the challenges of physical studies and improve the comparability of algorithms, the robotics community has turned to physics simulators. The advent of tools like Gazebo [2], OpenRave [3] and Mujoco [4] have given robotics and HRI researchers an avenue to test approaches before putting them in front of real users. And recently, simulation environments like AirSim [5] and OpenAI Gym [6] have allowed researchers to massively scale their simulations.

In response to challenges in HRI validation, we take inspiration from recent developments in scalable simulation to propose *HRI Gym*, an interface that facilitates the design and validation of human-robot user studies over the web.

We developed HRI Gym to facilitate four key components of HRI research: i) data collection from a wide group of users, ii) access to a variety of robotics and AI simulation environments, iii) a standardized library of HRI algo-

rithms, iv) and a framework for efficient evaluation of user studies. These elements and their dependencies are summarized in Figure 1.

Over the course of this paper, we propose a technical framework for launching user studies in the cloud (Section III) and show how the tool can be used for a specific application: imitation learning (Section IV). We also evaluate the tool via Amazon Mechanical Turk (MTurk) for various imitation studies (Section V), discuss the current limitations of our proposed HRI Gym and consider the challenges of moving local experiments to the web (Section VI).

II. Related Work

Over the last two decades, there have been numerous attempts to integrate robotics experiments with the web. In this section, we will discuss several approaches for collecting human data from the web for robotic tasks such as teleoperation of manipulators, collecting human demonstrations in manipulation and navigation, or shared autonomy.

Initial work in this area introduced the concept of *telerobots*, robots controlled by internet users that send commands usually via web browser [7–9]. Since then, the telepresence and teleoperation have been extensively studied in the context of robot-assisted surgery [10], hazardous task handling [11], elderly home care [12], and in various other domains [13]. Recently, open source tools like Robot Web Tools have helped communicate robotic information to the web, and systems like the Robot Management System have helped organize studies for teleoperation-based experiments [14, 15].

Teleoperating robots through web interfaces can help researchers collect a wide variety of user input for developing and validating algorithms such as learning from demonstration. In order to improve robotic grasping and manipulation, works such as [16] and [17] developed platforms for collecting user demonstrations over the web. Hu et al. have also focused on determining how to properly communicate between robots, servers, and users [18]. Most recently, the RoboTurk platform has been used to provide teleoperated demonstrations using MuJoCo-based [4] robot simulations [19]. While this project shares similar threads to HRI Gym, we specifically focus on providing a full pipeline for conducting HRI experiments including not only data collection but also standarized HRI algorithms and user surveying. Some of these previous web interfaces have also been used in conjunction with MTurk or other crowdsourcing platforms in order to collect input and feedback from web users with no previous robotics experience [20]. Our goal in this project is to bring together these functionalities physics-based simulated environments and integration with Amazon Mechanical Turk in one tool that enables efficient development and validation of HRI algorithms.

Web-based interfaces have also been adapted for HRI studies. Previous work has studied collaborative online games, where users control a human and robot avatar in order to generate behaviors for real robots [21–23]. Similarly, Tan et al. presents a method to teach collaboration from human pair demonstrations [24]. Another focus for web-based robotics has been crowdsourcing for Wizard of Oz studies. OzLab [25] and OpenWoZ [26] are tools that provide similar interfaces to researchers for demonstration tasks but with a focus on multimodal communication. Furthermore, recent work has begun to address how HRI can be combined with the Internet of Things to provide higher fidelity user input [27].

Previous work indicate the necessity and importance of a common platform that brings together HRI research under a single umbrella –integrating data collection from humans with a diverse set of robotics environments and library of HRI algorithms– and enables an easy design, evaluation and comparison of HRI studies and algorithms.

III. The HRI Gym Tool

The main goal of developing HRI Gym is to provide a general framework for online HRI studies. We aim to bring together different components necessary for developing and evaluating HRI studies under one common platform that can easily be modified and extended to include other techniques, environments, and algorithms by HRI researchers. HRI Gym is designed around the four components shown in Fig. 1:

1) **Data collection:** In any HRI study, the goal is to collect data and/or interact with a human user. As part of our tool, we facilitate large scale data collection from the browser to capture humans' input and feedback.
2) **Environment:** A server-based simulation component that simulates dynamic environments for robotics tasks, and provides an environment for interactions with humans.
3) **Algorithm:** A library of HRI algorithms that capture the human, robot, and environment data, and provide algorithmic results in the specified environment, e.g., reconstructed trajectories based on an imitation learning algorithm.
4) **Evaluation:** A framework to develop quantitative and qualitative evaluation and query humans' feedback for each experiment.

For instance, in a grasping study, human users would control a simulated robot arm (the environment) using

Fig. 2: Six key features that make up the core technical components of HRI Gym.

a joystick (the user input). The collected teleoperation data is then fed to an algorithm. As an example, a researcher could take demonstrated trajectories as an input to perform imitation learning algorithm. To validate the imitation learning algorithm, the reconstructed trajectories need to be validated by human users in the same environment. Our tool enables user studies that allow feedback (user input) enabling the designer to evaluate her hypotheses about the HRI algorithm (evaluation).

Usually, such human-subject studies are run locally. Since these components rely on local programs (simulators like Gazebo [2], Morse [28], OpenRave [3]), the human-subject studies can only be run locally. Today, researchers individually export their studies to the web, and collect data from Amazon Mechanical Turk. However, doing so requires them to individually rewrite each simulation program for the specific environments and algorithms they are interested in. Our tool standardizes this process. We emphasize that most HRI studies consist of these four components. In turn, the goal of HRI Gym is to provide a general framework for their integration. The six features that make up this framework, summarized in Fig. 2, are

1) **User Input:** To capture user input from the web, we provide methods to capture keyboard mouse, and evaluation form input over the web.
2) **Streaming:** To allow users to work in interactive environments, we provide methods to stream video from simulation, and to interact with server-based components.
3) **Repeated Algorithm Calls:** To run algorithms on collected data, we provide methods to call algorithms at different frequencies and convey user progress.
4) **Templating:** To move entire studies over to the browser, we provide pipelining tools and HTML templates for different types of user studies.
5) **Evaluation Design:** To allow users to evaluate final results, we provide methods to return visualizations and ask typical quantitative and qualitative questions.

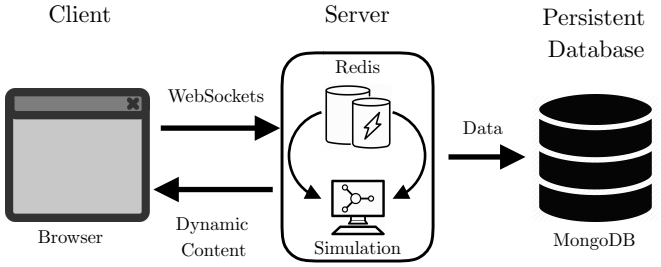


Fig. 3: A technical overview of how the HRI Gym application integrates its components. A client communicates with a server asynchronously and data about the experiment is saved intermittently to persistent storage.

6) **Web Application:** Finally, to improve usability, we organize the tool as a Python web application that is easy to install, configure, and scale.

In the course of developing HRI Gym, we sought to answer one main question: **Q1: Is the tool easy for users to use?** We will answer this question in the context of imitation learning in Section IV, but first we will explain the technical details of HRI Gym.

A. Technical Considerations

The general structure of user demonstration in HRI Gym is shown in Fig. 3. A web user connects to our server and reads static instructions about the experiment. Then, when the user begins the experiment, HRI Gym listens for their input in an asynchronous manner. Information about the user is stored locally and used to dynamically render a simulated environment, which is then streamed back to the user. Larger information about the experiment and user is saved intermittently to a persistent store.

In order to best facilitate different types of online HRI experiments, we focus on four technical issues: *1) Ease of Use, 2) Smart Storage, 3) Modular Human Input, 4) Minimizing Latency.*

1) Ease of Use: To allow for differences in user input, simulation, algorithms, and evaluation, we structure HRI Gym using a microservice paradigm.

The core application is built on Sanic[29], a Python web server. Since Sanic takes advantage of recent asynchronous features in Python, it enables us to quickly process method calls and scale applications while minimizing application overhead.

Each component is exposed as a set of API methods that can be called either from our pre-made HTML/CSS templates or from a different webpage. Since each service acts as a standalone program, we can easily integrate support for new inputs, environments, algorithms, and evaluations. This enables the designer to easily generate new HRI studies to evaluate a variety of HRI algorithms in supported robotics environments.

2) Smart Storage: Next, in order to efficiently communicate between components of an experiment, we utilize two stores of information: a high-throughput, *in-memory data store* and a slower, *persistent database*.

The in-memory data store is vital for interactive experiments where users directly interact with an environment. In these cases, the user input method has to continuously listen for inputs like key clicks and write to an in-memory data store, which is then used by the server-based simulation to update the user's environment. Though there are multiple options for in-memory databases, we chose to use Redis [30], a data store that prioritizes speed.

The persistent database, meanwhile, is used to store larger, more infrequent information about the experiment. We implement this using MongoDB [31], a popular NoSQL database.

3) Modular Human Input: On the client side, we provide support for keyboard, mouse, and form input through Javascript Websockets. We can also include video and voice based inputs using WebRTC [32], a framework for real time communication. The aim of this is to make it easy for researchers to listen for a certain type of user response without having to write their own methods for communicating human input from client to server.

4) Minimizing Latency: Given that most robotics simulation environments have to be run on server-side, our system mitigates the effects of latency as much as possible. This is done in part by choosing open-source components that are built to minimize server latency, such as Sanic and Redis. Then, on the simulation side, we silo each experiment and use environments that can scale across multiple processes.

IV. Application 1: Imitation Learning

In this section, we apply HRI Gym to design and evaluate a study for the problem of imitation learning (IL). The objective of imitation learning is to enable robots to acquire complex skills by observing demonstrations from a teacher. Often times, it can be quite tedious to program such behaviors with a manual set of rules [33]. On the other hand, it is far easier for a human expert to provide a *good* sequence of actions for a set of circumstances. Previous work in the fields of natural language and vision have shown that imitation of a given dataset can produce complex behaviors like dialogue and navigation [34].

A key impediment to imitation learning is the lack of large scale, high quality user demonstrations. Unlike standardized supervised learning problems, providing labels for robotics research often requires significantly more user effort. A common practice is to substitute the human expert with an agent trained via reinforcement learning (RL). Typically, a programmer designs a reward function and computes an approximately optimal policy using a standard RL algorithm [35]. This is then used to generate demonstrations that are then provided to the imitation learning module. There are several problems with this practice. Due to the difficulty in designing the reward function, the RL agent often does not capture the behavior we wish to learn [36].

This motivates the need for a tool that allows large scale collection of user demonstrations. We show here that HRI Gym is able to meet these requirements.

A. HRI Gym Framework for Imitation Learning

We now describe the concrete framework for running imitation learning studies using HRI Gym as illustrated in Fig. 4. This framework is for imitation learning in the *non-interactive* paradigm [37]. Here, the users provide a batch of demonstrations to the learning algorithm. The learning algorithm can only use this source of information to train a policy to imitate the user. The framework has the following four main components:

1) Collecting User Demonstrations: At the beginning of the study, the user is given instructions on what they are about to see, what the task is and the mapping from keyboard input to actions. They are then taken to a practice area where they can interact with the environment to familiarize themselves. In all of these environments, the user sees an image of the state at every time step. They enter an action for that time step by pressing a key and the observation-action pair is aggregated in a dataset.

2) Environments: The environment is a finite horizon Markov Decision Process (MDP) where an agent starts at an initial state, takes an action, transitions to a new state and proceeds until the episode terminates. To ensure the user has access to the full state of the MDP, we must be able to visualize the full state. To enable the user to provide controls using peripherals such as keyboards, mouses or joysticks, the environments must be able to accept some low-dimensional control. Finally the MDP must be tolerant to some amount of latency arising from

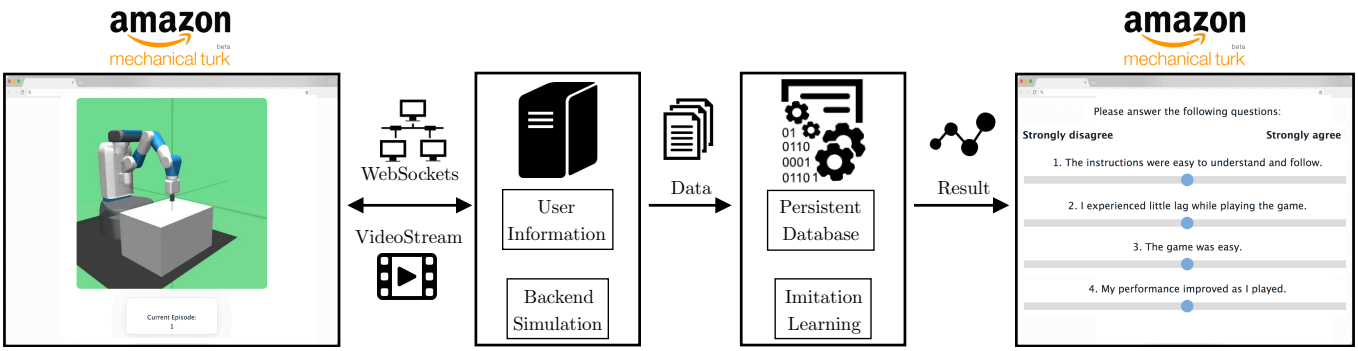


Fig. 4: The full pipeline used for imitation learning experiments. A user provides input for a server-based simulation then is asked to rate the trained agent and experience.

communication over the web. These three constraints restrict the types of MDP we can use.

In each of the environments we are to demonstrate, there is a reward function which is used by an RL algorithm. However, the human may not be optimizing for the same reward function. Since the imitation learning algorithms only rely on the actions provided by the human, it is not necessary that the learner would accrue a high reward.

We followed the conventions specified by OpenAI Gym [6]. Here, we choose four representative environments:

1) `FetchReach` (Fig. 5 (a)): The goal is to control the end-effector of a manipulator of a Fetch robot to reach a point in the 3D space. While this is quite simple for an RL algorithm, the fixed viewpoint and difficulty in control makes it challenging for a human user to solve within a time limit.

2) `FetchPush` (Fig. 5 (b)): The goal is to control the end-effector of a manipulator of a Fetch robot to push an object towards a goal. This is a classic non-prehensile manipulation problem which is more challenging for RL algorithms. The problem is solved only recently by a variant of RL algorithm that requires additional feedback from the environment [38].

3) `LunarLander` (Fig. 5 (c)): The goal is to land a lander at a specified location by operating its thrusters. Since the system is unstable and sensitive to latency, this is difficult for human users to directly control the thrusters.

4) `Montezuma Revenge` (Fig. 5 (d)): This is a classic Atari2600 game. Hence this is intuitive for human users and is quite difficult for RL algorithms due to its sparse reward landscape. For simplicity, we ask users to solve its first level only.

TABLE I: Questionnaire for evaluation of imitation learning interface

Questions
Q1. The instructions were easy to understand and follow.
Q2. I experienced little lag while playing the game.
Q3. The game was easy to use.
Q4. My performance improved as I played.
Q5. The experiment as a whole was enjoyable.

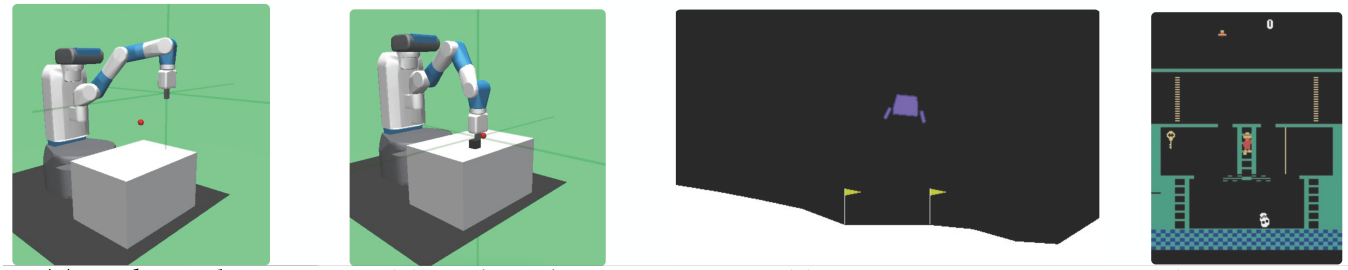
3) Imitation Learning Algorithms: To illustrate our tool, we consider a sub-class of imitation learning algorithms - *non-interactive model-free behavior cloning* [37]. In this subclass, the learner does not have an explicit model of the MDP. It can only sample transitions. The objective is to learn a policy, i.e. a mapping from states to actions. Since the setting is non-interactive, it can only use the dataset of demonstrated state-action tuples to do so.

We use a simple imitation learning algorithm - behavior cloning or supervised learning. This is a reasonable baseline to quickly ascertain the difficulty of an imitation learning task [39]. The format of data is designed to ensure that we can effortlessly replace this algorithm with other algorithms such as Generative Adversarial Imitation Learning (GAIL) [39].

4) Evaluation: We would like to leverage our tool to evaluate the implemented imitation learning algorithm in the environments presented. Specifically we wish to answer the main question raised in Section III in the context of imitation learning, i.e. whether the tool is useful for providing demonstrations and whether the demonstrations were useful for the user study. Hence, we derive the following set of hypotheses:

H 1. *Users are able to provide meaningful demonstrations using the tool.*

To test this hypothesis, we designed a questionnaire as shown in Table I. The user is asked to answer these questions on a Likert rating scale of 1-7.



(a) Fetch Reach (b) Fetch Push (c) Lunar Lander (d) Montezuma

Fig. 5: Four simulations that we used for imitation learning experiments. All of these are OpenAI Gym environments.

H 2. *The learner is able to achieve the same task performance as the user.*

This hypothesis tests the claim that while the learner may not be able to exactly imitate the user, the imitation is good enough with respect to the reward function accumulated by both. This can be tested by comparing the cumulative reward of the learner and the user.

H 3. *Imitating the user results in a better task performance than an RL agent.*

This hypothesis tests the utility of imitation learning in aiding reinforcement learning. If true, imitation learning can indeed be used to bootstrap reinforcement learning. This can be tested comparing the cumulative reward of the learner and the RL agent.

V. Evaluation of HRI Gym

Analysis of Imitation Learning Experiment. To answer the three hypotheses, we collected demonstrations for four environments from Amazon Mechanical Turk users following the procedure described above. In this section, we will summarize the collected user demonstrations and demonstrate how we can use them for imitation learning purposes.

A statistic summary of the data is presented in Table II. Depending on the length of the game, we hired different numbers of users to play a variable number of games. Each time a user played the game once, we obtained one demonstrated trajectory composed of a sequence of state and action pairs. At the end of each episode, the user received a certain reward, which was then used to compute the *success rate* for each experiment. Since imitation learning's performance relies on the quality of expert demonstration, we run imitation once with successful episodes and once with all the episodes.

In our evaluation, we mainly focus on two tasks: FetchReach and FetchPush. To evaluate the meaningfulness of user demonstrations as described in **H1**, we examine the users' success rates, which are the end effector being located within a range of the goal for the FetchReach task, and an object reaching a range of the goal position for the FetchPush task, as shown in Fig. 6a and 7a. Though users are likely to fail to solve the FetchReach task, they demonstrated a significantly higher success rate on FetchPush than trained DDPG agents. The success rates along with users qualitative responses about the tool as shown in Fig. 8 validate our first hypothesis. We did not conduct imitation learning on the Montezuma's Revenge task; however, previous work has shown successful imitation on the environment when combining imitation and reinforcement learning [40, 41].

To examine whether the imitator can achieve the same task performance as humans as described in **H2**, we present the results of imitation on user demonstration on the FetchReach and FetchPush tasks. For each environment, we utilize two versions of the imitator, the first of which learns from all the collected data and the second of which uses only selected successful episodes. We label this second agent the selective imitator. We then compare their performances against an RL agent trained using DDPG to answer **H3**. Fig. 6a and 7a show the success rate of each agent along with their training iteration. Fig. 6b and 7b show the success rate of each agent along with the number of observation-action pairs it observes.

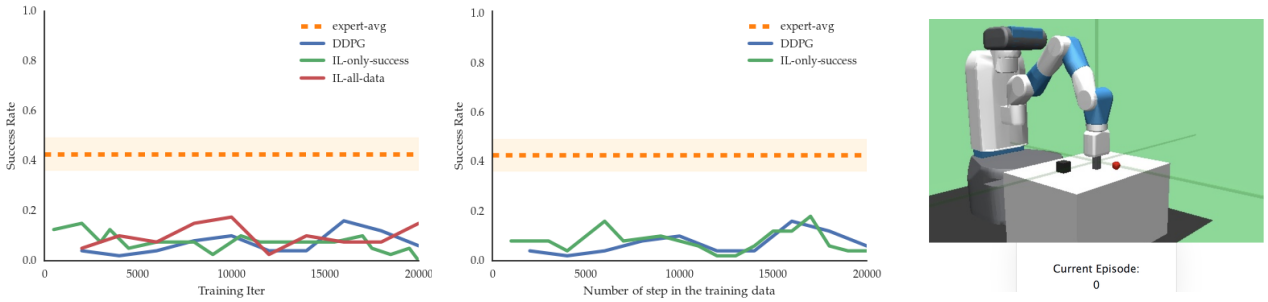
From Fig. 6a, we observe that a selective imitator could achieve a similar success rate as the human users on FetchReach whereas an indiscriminate imitator achieve less successful results. However in Fig. 7a, both imitators perform poorly compared with the users, regardless of having more data than FetchReach. Given that FetchReach has a 13-dimensional state space while FetchPush has 28-dimensional state space, the performance of imitator is likely to be subjective to the complexity of the environment. Interestingly, for both environments, the amount of user data does not show a significant amount of impact on its performance, as shown in Fig. 6b and 7b. As we have demonstrated through these two tasks, in one case the learner is able to achieve similar performance as the user, and in the other the learner is not capable of doing so. This validates the importance of having a tool where the designer can easily swap different environments to validate the same implementation of an

TABLE II: Statistics of collected users demonstrations

	# users	# trajectories	# steps	trajectory success rate (%)	# steps in success trajectories
FetchReach	10	75	13049	25.33	1849
FetchPush	10	220	70715	42.51	20315
LunarLander	65	100	11282	31.00	2638
Montezuma's Revenge	7	7	16486	85.71	6486

(a) FetchReach success with training (b) FetchReach success with data (c) FetchReach environment

Fig. 6: Evaluation on the FetchReach environment



(a) FetchPush success with training (b) FetchPush success with data (c) FetchPush environment

Fig. 7: Evaluation on the FetchPush environment

algorithm. This imitation learning example shows that the same implementation of the algorithm in two different environments (FetchReach and FetchPush) can result in different responses to answering **H2**.

Can imitation learning lead to a better task performance than a trained RL agent? This experiment might be insufficient in answering this question, as none of the imitators we have outperformed an RL agent. Nevertheless, it can be the key to a new stage of imitation learning. Many imitation learning algorithms have proved its efficiency in imitating a well-trained or carefully-crafted RL agent of rule-based agent. Our results suggested that such imitation learning algorithm might be unfit when it comes to learning from human's demonstration.

User Likert Ratings. To understood how end users reacted to experiments created with HRI Gym, we asked 30 users to rate their experience with the tool, 10 with a Fetch Reach environment, 10 with a Lunar Lander environment, and 10 with an Atari environment. Each would rate their agreement with the statements in Table I, on a scale of 1 to 7. The results of this user study are summarized in Fig. 8. Across all three tasks, users generally rated the instructions as clear and understandable. Users reported relatively similar amounts of lag

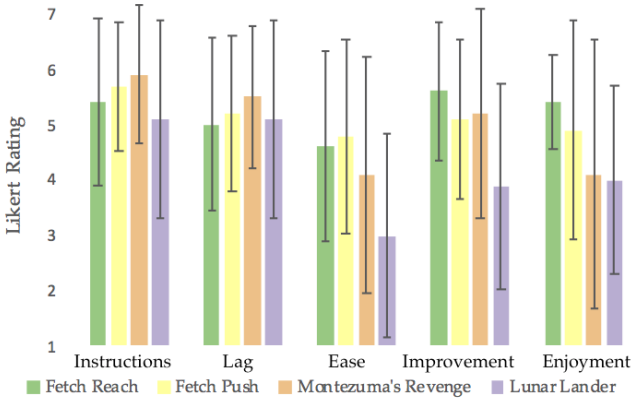


Fig. 8: User ratings corresponding to the questions in Table I for four different experiments. Survey size = 10 users for each experiment.

for all three environments, while the most dynamic task, Montezuma's revenge had the highest score for latency. In the Lunar Lander environment, users found it difficult to control the lander just with its thrusters and improved their skills, as shown in responses to questions 3 and 4. This was also partially true for the Montezuma's Revenge environment, which involved users having to combine multiple actions, like jumping and moving at the same time. Finally, users expressed a variety of scores for enjoyability, with user ratings being most divergent for the Atari task. This further points to user engagement being determined by specific environments, rather than through the framework they are represented in. Such qualitative measures gives us more insight into future experiment designs using HRI Gym.

We have also asked for additional comments, the majority of users commented on the quality of input they were providing. Though they found the system responsive, some found issue with the intuitiveness of control in some environments, especially in the Lunar Lander task, where the thruster control is not as intuitive. This further shows the importance of designing environments that are easily controllable.

Overall, this user data analysis gives important insight into our initial questions **Q1**: is the tool easy for users to use? Though users found it difficult to provide demonstrations in some environments, the quality of demonstration shown in Section IV and generally positive responses across all four environments pointed to the tool being successful among end users. Therefore, we found that **Q1** is not always true, but ease of use depends overwhelmingly on the quality of environment being tested.

VI. Discussion

We have presented an online tool to run human-robot interaction user studies over the web and presented a successful use case of it in imitation learning. Looking forward, we would like to discuss the flexibility of HRI Gym, its current limitations, and plans for future work.

Tool Flexibility. The main goal of HRI Gym is to enable users to seamlessly launch user studies on the cloud. While we have only presented the specific example of imitation learning, we would like to reiterate that HRI Gym can be used for many other user experiments such as producing legible motion [42, 43], inverse reward design [36], preference learning [44], intent inference [45], and any other HRI algorithm that can be evaluated in a simulation environment.

Technical Limitations. There are a number of technical issues to be aware of when porting experiments to HRI Gym. One such issue is latency. Since running environments on the client side is restrictive and time-consuming, researchers will most likely have to host experiments on servers. Consequentially, when experiments are run from servers, the designer will have to deal with both user latency, i.e., the time it takes to send information between the user and the server, and server latency, i.e., the time it takes to process information within the server. In our imitation example, we aimed to mitigate these by restricting how far a user could be from our server and using compute-effective `Box2d` and `Mujoco` environments. However, the success of a HRI Gym experiment still very much depends on the specific environment and input used.

User Limitations. We also have to address issues that arise from obtaining inputs from users over the web. When studies are crowd-sourced, how can researchers deal with variation in user demonstrations? How can environments be made simple enough to allow effortless user interaction, but also complex enough to thoroughly study an algorithm or phenomenon? How can we help researchers deal with suboptimal user demonstrations? These questions must be answered in the context of specific experiments.

In our study of imitation learning, we found that it was difficult to collect user demonstrations in environments with unstable dynamics and high dimensional control inputs, meaning experiments either had to be scaled down or abandoned. Interestingly, these issues were discovered only by examining the data from the study. Hence the hope is that, given access to a larger database of test users, HRI Gym will allow researchers to quickly identify issues and roll out modifications to end-users.

Future Work. We plan to continue developing the HRI Gym tool to facilitate as many types of experiments as possible by adding support for many more inputs, environments, algorithms, and evaluations. Although we have only used keyboard input thus far, we plan to include sound, video, and text inputs. We also hope to integrate tools like Nvidia Capture and adaptive streaming to more efficiently stream information to study participants' behaviors.

Furthermore, although our example studies were built off simulated Gym environments, which already have well-documented script render and input methods, we aim to show our tool working on unsupported computer simulations (i.e. Gazebo [2] and OpenRave [3]) and on real-world robots. To facilitate these, we plan to build an extension that allows simulations to run off separate server instances.

Finally, although the first version of HRI Gym was designed to have single user studies, we are actively working to add multi-user support to enable experiments on collaborative planning and shared autonomy.

References

[1] Michael A Goodrich, Alan C Schultz, et al. Human–robot interaction: a survey. *Foundations and Trends® in Human–Computer Interaction*, 1(3):203–275, 2008.

[2] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, 2004.

[3] Rosen Diankov and James Kuffner. Openrave: A planning architecture for autonomous robotics. Technical report, 2008.

[4] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[5] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *CoRR*, abs/1705.05065, 2017. URL http://arxiv.org/abs/1705.05065.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[7] D. Wright, Campbell Rogers, Daniel I. Simon, Zhiping Chen, Tatsuya Fukutomi, Re C. Zago, Raila Ehlers, and Patricia A. Detmers. A telerobot on the world wide web. In *National Conference of the Australian Robot Association*, 1995.

[8] Raúl Marín, Pedro J. Sanz, Patricio Nebot, and Raul Wirz. A multimodal interface to control a robot arm via the web: a case study on remote programming. *IEEE Transactions on Industrial Electronics*, 52:1506–1520, 2005.

[9] Ken Goldberg, Joseph Santarromana, George Bekey, Steven Gentner, Rosemary Morris, Jeff Wiegley, and Erich Berger. The telegarden. In *Proc. of ACM SIGGRAPH*, pages 135–1140, 1995.

[10] Paul Vespa. Robotic telepresence in the intensive care unit. *Critical care*, 9(4):319, 2005.

[11] Woosub Lee, Sungchul Kang, Munsang Kim, and Mignon Park. Robhaz-dt3: teleoperated mobile platform with passively adaptive double-track for hazardous environment applications. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 33–38. IEEE, 2004.

[12] Patrick Boissy, Hélène Corriveau, François Michaud, Daniel Labonté, and Marie-Pier Royer. A qualitative study of in-home robotic telepresence for home care of community-living elderly subjects. *Journal of telemedicine and telecare*, 13(2):79–84, 2007.

[13] Kevin K Chung, Kurt W Grathwohl, Ron K Poropatich, Steven E Wolf, and John B Holcomb. Robotic telepresence: past, present, and future. *Journal of cardiothoracic and vascular anesthesia*, 21(4): 593–596, 2007.

[14] Russell Toris, David Kent, and Sonia Chernova. The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing. *Journal of Human-Robot Interaction*, 3(2):25–49, 2014.

[15] Russell Toris, Julius Kammerl, David V Lu, Jihoon Lee, Odest Chadwicke Jenkins, Sarah Osentoski, Mitchell Wills, and Sonia Chernova. Robot web tools: Efficient messaging for cloud robotics. In *IROS*, pages 4530–4537, 2015.

[16] Ben Kehoe, Akihiro Matsukawa, Sal Candido, James Kuffner, and Ken Goldberg. Cloud-based robot grasping with the google object recognition engine. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4263–4270. IEEE, 2013.

[17] Alexander Sorokin, Dmitry Berenson, Siddhartha S Srinivasa, and Martial Hebert. People helping robots helping people: Crowdsourcing for grasping novel objects. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2117–2122. IEEE, 2010.

[18] Guoqiang Hu, Wee Peng Tay, and Yonggang Wen. Cloud robotics: architecture, challenges and applications. *IEEE network*, 26(3), 2012.

[19] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

[20] Ken Goldberg and Ben Kehoe. Cloud robotics and automation: A survey of related work. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-5*, 2013.

[21] Sonia Chernova, Nick DePalma, Elisabeth Morant, and Cynthia Breazeal. Crowdsourcing human-robot interaction: Application from virtual to physical worlds. In *RO-MAN, 2011 IEEE*, pages 21–26. IEEE, 2011.

[22] Nick DePalma, Sonia Chernova, and Cynthia Breazeal. Leveraging online virtual agents to crowdsource human-robot interaction. In *Proceedings of CHI Workshop on Crowdsourcing and Human Computation*, 2011.

[23] Cynthia Breazeal, Nick DePalma, Jeff Orkin, Sonia Chernova, and Malte Jung. Crowdsourcing human-robot interaction: New methods and system evaluation in a public environment. *Journal of Human-Robot Interaction*, 2(1):82–111, 2013.

[24] Jeffrey Too Chuan Tan, Yoshinobu Hagiwara, and Tetsunari Inamura. Learning from human collaborative experience: Robot learning via crowdsourcing of human-robot interaction. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 297–298. ACM, 2017.

[25] John Sören Pettersson and Malin Wik. *Perspectives on Ozlab in the cloud: A literature review of tools supporting Wizard-of-Oz experimentation, including an historical overview of 1971-2013 and notes on methodological issues and supporting generic tools*. Karlstads universitet, 2014.

[26] Guy Hoffman. Openwoz: A runtime-configurable wizard-of-oz framework for human-robot interaction. In *2016 AAAI Spring Symposium Series*, 2016.

[27] Béni-Trésor Akimana, Maxim Bonnaerens, Jonas Van Wilder, and Bjorn Vuylsteker. A survey of human-robot interaction in the internet of things.

[28] Gilberto Echeverria, Séverin Lemaignan, Arnaud Degroote, Simon Lacroix, Michael Karg, Pierrick Koch, Charles Lesire, and Serge Stinckwich. Simulating complex robotic scenarios with morse. In *SIMPAR*, pages 197–208, 2012. URL http://morse.openrobots.org.

[29] Sanic web server, Oct 2018. URL https://github.com/huge-success/sanic.

[30] Jing Han, E Haihong, Guan Le, and Jian Du. Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pages 363–366. IEEE, 2011.

[31] Kristina Chodorow. *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. " O'Reilly Media, Inc.", 2013.

[32] Alan B Johnston and Daniel C Burnett. *WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web*. Digital Codex LLC, 2012.

[33] J. Andrew (Drew) Bagnell. An invitation to imitation. Technical Report CMU-RI-TR-15-08, Carnegie Mellon University, Pittsburgh, PA, March 2015.

[34] Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. Talk the walk: Navigating new york city through grounded dialogue. *CoRR*, abs/1807.03367, 2018.

[35] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[36] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems*, pages 6765–6774, 2017.

[37] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and TrendsÂ® in Robotics*, 7 (1-2):1–179, 2018. ISSN 1935-8253. doi: 10.1561/2300000053. URL http://dx.doi.org/10.1561/2300000053.

[38] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[39] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[40] Tobias Pohlen, Bilal Piot, Todd Hester, Mohammad Gheshlaghi Azar, Dan Horgan, David Budden, Gabriel Barth-Maron, Hado van Hasselt, John Quan, Mel Večerík, et al. Observe and look further: Achieving consistent performance on atari. *arXiv preprint arXiv:1805.11593*, 2018.

[41] Yusuf Aytar, Tobias Pfaff, David Budden, Tom Le Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by

watching youtube. *arXiv preprint arXiv:1805.11592*, 2018.

[42] Anca Dragan and Siddhartha Srinivasa. Generating legible motion. 2013.

[43] Anca D Dragan, Kenton CT Lee, and Siddhartha S Srinivasa. Legibility and predictability of robot motion. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 301–308. IEEE Press, 2013.

[44] Anca D Dragan Dorsa Sadigh, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems (RSS)*, 2017.

[45] Shervin Javdani, Yuxin Chen, Amin Karbasi, Andreas Krause, Drew Bagnell, and Siddhartha Srinivasa. Near optimal bayesian active learning for decision making. In *AISTATS*, 2014.