BC+RL: Imitation Learning From Non-Optimal Demonstrations

Jonathan Booher [*1]

Abstract

Reinforcement Learning approaches in robotics are difficult because of the lack of an easy to define reward function that is dense in time. This suggests the use of imitation learning as a way to meaningfully guide exploration in an intelligent manner. We present an algorithm that allows for successful imitation learning from suboptimal demonstrations by combining behavioral cloning approaches with pure reinforcement learning to accelerate learning from sparse reward functions in robotic domains with long time horizons. For code: click here

1. Introduction

Reinforcement learning (RL) has shown great ability to solve complex tasks in many different domains. However, the utility of RL algorithms and approaches to meaningful robotics tasks is hindered by the fact that in many cases, it is not possible to design reward functions that are dense in time and as a result so-called "sparse" rewards functions are needed. This poses a unique challenge for long time horizon tasks as the exploration task is made difficult. Recent approaches (Riedmiller et al., 2018) have shown advances in dealing with sparse rewards; however, the difficulty suggests the use of imitation learning to aid in exploration such as the usage of reverse curricuums like (Fan et al., 2018) and (Salimans & Chen, 2018). Many approaches in imitation learning require near optimal actors, but gathering optimal demonstrations is expensive and typically involve expensive Virtual Reality equipment which makes such approaches cost and time prohibitive to collect large volumes of data for imitation learning algorithms. This limitation was addressed in (Mandlekar et al., 2018) which created a system that is able to efficiently and cheaply croudsource robotic trajectories in simulated environments as well as on real robots. This recent advance motivates the creation of imitation learning algorithms that benefit greatly from large quantities of data that may not be optimal. One the approaches to imitation learning that benefits from large amounts of data is behavioral cloning, but this approach cannot exceed the performance of the actors that collected the trajectories and it is unclear how well pure behavioral cloning works on highly multi-modal data from multiple different actors. Pure RL, on the other hand, is capable of achieving high rewards on environments.

We seek to address these shortcomings. Thus, the proposed approach is a combination of these two approach. The approach uses a BC algorithm to guide the exploration of an RL algorithm by providing the policy with a good initialization and additional supervision throughout training. However, since a policy is being trained using RL, we can exceed the performance of the actors that collected the demonstrations.

2. Prior Work

In the study of imitation learning, many works like (Zhang et al., 2017) have used VR equipment to collect high quality demonstrations but only on the order of a couple hundred. Other works, like RoboTurk use widely available hardware (iPhones) to collect large amounts of trajectories that are not optimal because of lower fidelity in the tracking and depth perception from the operators. That work showed that while certain segments of the trajectories may be suboptimal, there is not a significant difference in completion time between their control and that of VR setups. Further works that leveraged the RoboTurk data, for example the scalable RL framework Surreal (Fan et al., 2018) showed usage of PPO from demonstrations by resets to demonstration states, but this is an inefficient use of the collected data as it is only incorporated in the initial state of the world.

Some work has used behavioral cloning methods to initialize other imitation learning algorithms like generative adversarial imitation learning (GAIL) (Ho & Ermon, 2016). However, in these methods, the data is discarded after the initial pretraining of the network. Other works have combined GAIL with PPO to be able to combine the benefits of RL and Imitation Learning (Zhu et al., 2018); however, these methods are difficult to train as GAIL can suffer from mode collapse or unstable training. It is these last two works that serve as the motivation for this work.

3. Task Description

We approach the the task first in a simulated environment of a Sawyer robotic arm with a single object (a block) that

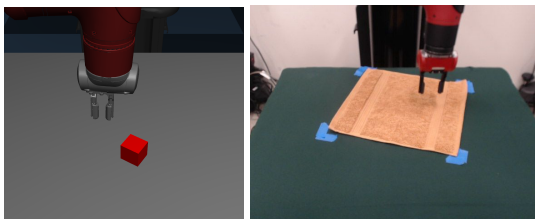

Figure 1. **Left:** An image from the $SawyerLift$ environment in robosuite. The goal is to lift the cube a certain distance off the table. **Right:** An image of the task in reality. The task goal is to manipulate the cloth grasping the cloth to lift it or unfold a partially folded cloth.

needs to be lifted a certain distance off of the table. This environment is created in $robosuite$ which is a part of (Fan et al., 2018) and is called $SawyerLift$. We then test on real world data collected on real versions of the Sawyer arms and on a similar task of manipulating a deformable towel by lifting the towel off the of the table from the corner, a subtask for a complex unstructured task of unfolding a crumpled cloth.

4. Approach

The proposed approach is a combination of Behavioral Cloning and Reinforcement Learning to address the pitfalls of each. Reinforcement Learning for long horizon tasks struggles with sparse rewards, and behavioral cloning requires optimal demonstrations. To address these issues, we augment the loss of an arbitrary reinforcement learning agent with a behavioral cloning loss. Furthermore, we have the total loss be a convex combination of the RL loss and the BC loss where early in training, the weight of the BC loss is 1 and late in training, the weight is 0. Therefore, the loss being optimized is

$$\theta_t L_{BC} + (1 - \theta_t) L_{RL}$$

$$\theta_t = \begin{cases} 1 & t < \alpha \\ exp(-\frac{t}{\beta}) & \alpha \leq t \leq \phi \\ 0 & \phi < t \end{cases}$$

where α, β, ϕ are all hyperparameters. This schedule exponentially decays the BC loss over time after the BC loss gives the network a decent initialization until a certain amount of time has elapsed and then we gradually phase in RL so that we can eclipse the performance of the demonstrators. This approach is agnostic to the RL algorithm being employed and simply requires that the network used for behavioral cloning is the same as that of the reinforcement learning algorithm. For the purposes of this work, we are concerned with the following behavioral cloning loss

$$w_1 l_1 + w_2 l_2 + w_g l_g + w_{cos} l_{cos}$$

$$l_1 = ||\pi(s) - a||_1, \;\; l_2 = ||\pi(s) - a||_2$$

$$l_g = cross\_entropy(\pi_g(s) - a_g)$$

$$l_{cos} = cosine\_distance(\pi(s), a)$$

where each w is a weighting of the loss. We use l_1 and l_2 to calculate distances between the actuations predicted and the actual taken actuations. However, these losses encourage exact matching of the actions, however, in the context of the task we are examining, it makes more sense to encourage movement *in the same direction* which suggests the inclusion of the cosine distance as another additional loss. The cross entropy is used as a classifier on gripper state, this is needed as there will be many more states in which the gripper will be open than closed in the data collected. Further additional losses can be used when training from pixels like predicting the location of object in the scene, but in the case of robotics in the real world, this is, more times than not, information that is unavailable. We can however, add in auxiliary losses for the location of the end effector as well as a predicted grasp location. We can gather the ground truth for grasp location easily by examining the trajectories and taking the end effector location when the gripper is actuated. The location of the end effector is information that can be gathered from real robot demonstrations so it is applicable in this context. We realize these losses as follows:

$$z = CNN(x_{image}), \quad ss = spatial - softmax(z)$$

$$l_{eef-location} = ||gt_{eef-location} - MLP(ss)||_2$$

$$l_{grasp-location} = ||gt_{grasp-location} - MLP(ss)||_2$$

where gt_* is the ground truth information. We additionally take the output of the spatial-softmax layer as input to the policy MLP. Adding these additional loss terms greatly improves convergence as it forces a rich latent space representation of the images. Empirically, the more important auxiliary loss is the localization of the current end effector location. We additionally note the interesting property that we could use these two predicted locations (if they are accurate enough) to run a planning algorithm instead of training a policy. During the experiments, we find that while the network is able to effectively minimize these auxiliary losses, they are not accurate enough for planning.

We additionally add another loss focusing specifically on rotation and is used as a measure of similarity between the predicted delta in orientation and the actual delta in orientation this is needed because initial experiments showed a lack of correlation between small absolute errors in orientation and actual rotational error. This has the added benefit of speeding up training. As part of this loss, we add a term that penalizes changes in sign as there are two quaternions that represent the same rotation but in different orientations, as we are operating in deltas, this would be an issue when it comes to running on the real robot.

5. Experiments

5.1. Policy Structure

We seek to learn policies that leverage visual information as well as information about the internal robot state. This will allow the image portion of the network to focus on understanding the environment and creating goals for the agent. Thus, our policies are structured as follows

$$s = CNN(x_{image}) \rightarrow MLP([s, x_{proprioception}])$$

this will allow us to fuse image and low dimensional state information to ease the task of learning as we make the environment more observed. We chose to output a single fully connected layer from the MLP that contains information about the delta control and the gripper actuation. We chose to output deltas in position as that allows for a direct use of the dataset gathered by the RoboTurk system that uses deltas and position and deltas in rotation to control the robot in end effector space. This also eases the challenge of policy learning on the real robot as in the case of joint level control, multiple different joint positions are possible to reach the same end effector point in space with a given orientation. In end effector control, we leave the direct calculation of the joint angles to an IK solver to remove this ambiguity. Furthermore, this allows us to use the same controller as the one that the data was collected with in the first place, which is good for behavioral cloning approaches. The exact structure of the network can be seen in Fig 2 including predictions for auxiliary tasks described later. When combining behavioral cloning with PPO, we use the output of the network to predict the end effector actions and treat the output of the network as the means for the sample distributions for PPO.

5.2. Inverse Kinematics

Controlling robots using delta positions and orientations at the end effector poses the challenge that the methods of control of the robots are not in deltas at the end effector but rather controls at the joints. This presents the issue of inverse kinematics (IK) which is to map end effector positions to joint angles, velocities. We chose to work with directly setting joint angles in simulation and controlling using velocities on the real robot. We use Bullet IK to calculate the joint positions for a given delta in position and orientation and constrain the deltas to be in fixed ranges ($||delta\_position|| \leq 0.05$, $||delta\_rotation|| \leq \pi/40$) so that the velocities and accelerations of the joints remain in feasible ranges. For the real robot, we use a proportional control so that the effort exerted by the robot is proportional to the distance the joint positions are away from the target joint positions from the IK solution.

5.3. Learning Algorithm

We use an implementation of PPO to learn policies because of its property of discouraging large changes in the policy per update, PPO has been widely used in robotics when learning on real robots for the sake of robot and human safety. In this way, PPO is related to TRPO, the policy is discouraged from varying by a lot during one step of the optimization. But while TRPO requires expensive computation of the Fischer information matrix to ensure optimization within a trust region, PPO adds a differentiable penalty for large policy changes. In particular, we use a clipped surrogate loss function:

$$r_t(\theta) = \frac{\pi_t(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

$$L^{CLIP} = E[\min\{r_t(\theta)A_t, clip(r_t, 1-\epsilon, 1+\epsilon)A_t\}]$$

In this formulation, r is an importance sampling step, A_t is the calculated advantages, ϵ is a hyperparameter, typically 0.2 and the expectation is over samples. This loss replaces the policy gradient and encourages the policy gradient to have a lower magnitude. There is an additional value function loss which is simply MSE

$$L^{VF} = MSE(\ MLP(s),\ R)$$

where R is the returns and the MLP is a single layer network from the latent space of the main policy network (ie the network architecture has two heads, one for policy and the other for value) with shared weights further up the network. The weight sharing allows us to use the value difference to augment the loss of the policy gradient to help with convergence. We can then combine the loss terms as follows

$$L^{PPO} = c_1 \cdot L^{CLIP} - c_2 \cdot L^{VF}$$

where c_1, c_2 are weighting coefficients. In our experiments, we use $c_1 = 1.0$, $c_2 = 0.5$. In the literature, there is sometimes a penalty for the entropy of the policy distribution, but this is largely for discrete action spaces that is not applicable in our continuous control space.

Our implementation of PPO was verified by training a policy on a simpler environment ($HalfCheetah$) before experiments were conducted on the more complicated 6-DOF manipulation environment. We also leverage the Distributed PPO implementation from (Fan et al., 2018) to quickly learn a robust policy in simulation that achieves near optimal performance on the relatively difficult $SawyerLift$ task.

6. Results

We begin the experiments in simulation to verify that the BC implementation as well as the BC+RL algorithm are effective before attempting to extend the method into the real world.

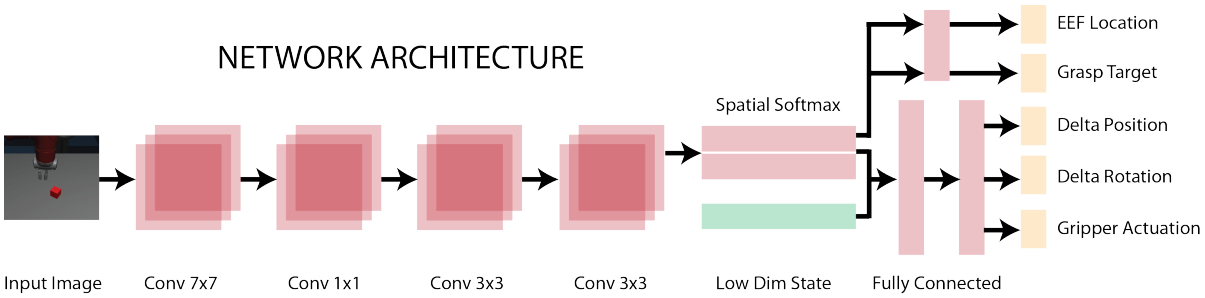

Figure 2. The architecture of the network used. We take input images of size $128x128x3$ and run them through a set of convolutional layers that serve as feature extractors for the image. We then use the spatial softmax layer to extract the locations of the features and then proceed to two heads. The first head is the auxiliary prediction head, we use these to encourage a rich feature representation to be learned by the convolutional network encoding useful information like predicted grasp points and the current end effector location. The other head is the policy head which uses fully connected layers to predict delta poses of the robot end effector. It is off of this head that the value function branches off.

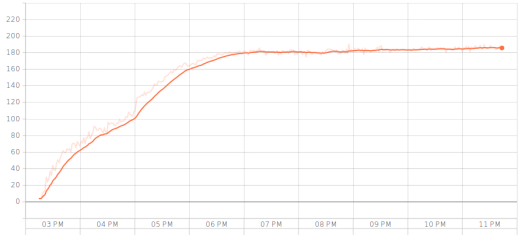

Figure 3. Sum of environment rewards for PPO agent on $SawyerLift$ over time. Note the maximum possible reward is 450 which corresponds to instantaneous solving of the task. The time horizon is 200 and there is no discount factor. This policy was trained using end effector controls (delta position and delta orientation in quaternion). Ideal policies attain a return of 400 discrepancies can be attributed to inaccuracies in IK control.

6.1. Expert Policy and Demonstration Generation

We first create an implementation of PPO to learn policies using the fusion of image and low dimensional states. This implementation is a vanilla implementation of PPO that is able to achieve a return of 100 over the span of a 200 timestep episode. Note that the maximum possible return over this episode length is 450. This is low, but is in accordance to the results reported in previous works specifically applying the Distributed PPO method of (Fan et al., 2018) with exactly one actor (see figure 10 of (Fan et al., 2018)). To generate the episodes that we use to perform the BC segment of our method, we train a policy using Surreal distributed PPO using end effector control with 8 actors to achieve a policy that attains an average sum of rewards of 180 which differs from the results from the Surreal paper which was able to train a policy to attain a reward of close to 400 (see fig 3 for or learning curve). Potential reasons for this discrepancy will be discussed later. the policy that we trained took over 10 hours to fully train to the stated performance.

We collect demonstrations from this trained policy by perturbing the policy's selected actions by adding gaussian noise with 0 mean and standard deviation ranging from 0 to 0.1. This creates policies with varying levels of success and varying levels of optimality allowing us to test the method using demonstrations that are of different qualities.

6.2. Behavioral Cloning in Simulation

Next, we evaluate the pure BC approach on the gathered trajectories to evaluate if BC will be capable of giving PPO a good initialization. We observe difficulty in effectively learning to replicate the deltas in orientation of the policy. To address this, we introduce another loss term that is a special distance metric for quaternion rotations

$$1 - |\hat{q}^T q|$$

where $\hat{q}$ is the predicted quaternion and q is the ground truth quaternion rotation and $|\cdot|$ is the absolute value. This approach is commonly used in pose estimation tasks as a loss specific for object rotation (Bousmalis et al., 2016), note that this loss is agnostic to the two different orientations of "covers" of the quaternion rotation. Further experiments suggest the model tends to overfit on trajectories and as such $l2$ weight decay is added in conjunction with ADAMW optimization to better handle the decay.

We observe rapid convergence in simulation on noisy demonstrations that maintains the ability for the BC policy to generalize to held out trajectories. We evaluate the BC policy on held out trajectories as well as directly rolling out the BC policy that is learned. On the held out trajectories, we achieve error in delta position of approximately 6 degrees per time step (see Fig 4) and a low absolute difference on the order of $1e-4$. As for the deltas in orientation, the quaternion loss allows us to reach an average error in orientation of $3e-2$ radians (See Fig. 5).

Noise std	Average Return	% Recovered by BC
0.0	180	85
0.01	173	67
0.1	169	83

Table 1. Evaluation of behavioral cloning method for various different scales of per action noise on demonstrations. The higher noise dataset performs well as the noise helps BC address the compounding error problem by providing a "cone" of trajectories.

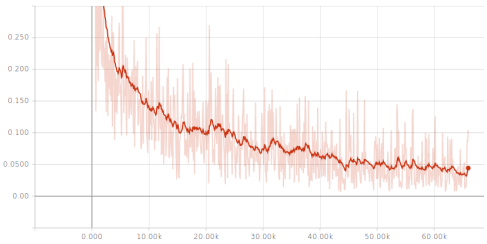

Figure 4. Cosine error in delta position prediction over time

We then test with rollouts of the BC policy and see that we can recover the majority of policy performance through pure BC given relatively few demonstrations as well as in a short period of time. In the randomized environment where the position of the cube is randomized, our results for rollouts of the BC policy are in Table 1

These results show that pure BC is not able to fully handle the task on new rollouts, this is largely due to the randomization of the environment (which BC as a supervised learning algorithm is brittle toward) as well as the relatively few trajectories given to the algorithm. Additionally, the BC method runs into the possibility of compounding errors when it finds the robot in a pose that the network has not generalized to account for.

We note that training a successful RL policy using end effector deltas proved difficult and unreliable. The IK controller that we use is based on Bullet IK, but the IK methods from Bullet have a memory leak in them which forced a workaround of periodically restarting agent processes when memory consumption became an issue. This potentially has adverse effects on learning. Additionally, the IK is not perfect, with a given desired delta in position, the robot is not guaranteed to attain that position, there is some non-determinism that exists in the form of numerical inaccuracies. We attempted to train a policy using joint level commands which was able to attain a return of 400 and extract delta end effector poses from it, but these proved difficult to learn from because of the IK inaccuracies. However, the success of the method on held out trajectories and summing the errors over an entire trajectory led to good results which suggests the inaccuracies are rooted, in large part, in the IK computations.

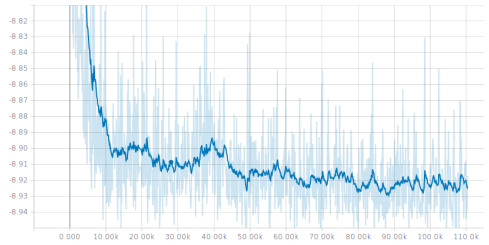

Figure 5. Error in delta rotation prediction over time. Note the screenshot was taken at a later point in training.

Noise std	Average Return	% Recovered by BC+RL
0.0	180	95
0.01	173	87
0.1	169	92

Table 2. Evaluation of combined behavioral cloning and reinforcement learning method for various different scales of per action noise on demonstrations.

Even though we were not able to replicate an ideal policy on this environment, the results are positive as they show that it is indeed possible for BC to provide a useful initial policy and further guidance (in the form of additional supervision) to an RL algorithm.

6.3. Behavioral Cloning + Reinforcement Learning in Simulation

We now proceed to evaluate the combined BC+RL algorithm. We use a normal, single agent implementation of PPO using sahred weights with the BC model. We augment the loss function with the scheduled weighting described in the method section.

Particularly, we use an exponential decay schedule with parameter 0.97 for the decay and 10000 iterations of optimization before the decay begins, with a decay every 100 optimization where the schedule is a step function). This method is able to almost fully replicate the original policy in a shorter period of time, all BC+RL experiments were run for 7 hours because of time constraints. Again, some of the inaccuracies are likely attributable to the IK not being able to attain the control that was being applied. Results from training the full algorithm can be seen in Table 2. We see similar convergence properties for the components of the BC losses during training of the combined algorithm (see Fig 6) and, we also see similarly low errors on held out trajectories compared to when training the pure behavioral cloning algorithm.

These results again indicate possible errors within the IK controller, but suggest that the combined algorithm is capable of surpassing the performance of the BC algorithm and approaching the performance of the original expert policy.

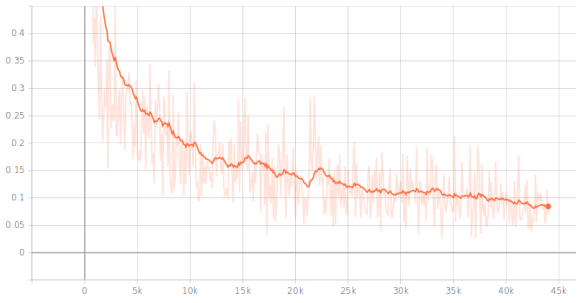

Figure 6. Graph of the cosine distance loss over time during training the BC+RL algorithm. This loss is the most pertinent to the tasks that are being examined in this work. Note the similar convergence properties as compared to Fig 4

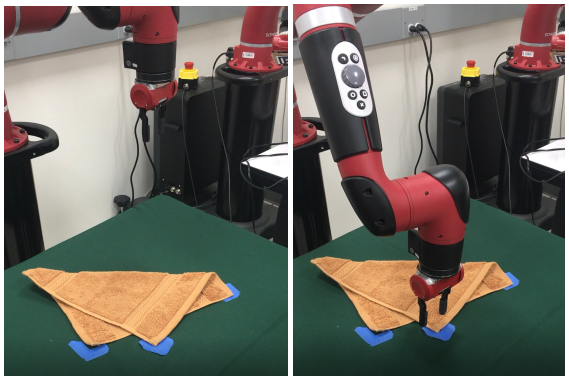

Figure 7. **Left:** The initial configuration of the robot. **Right:** An example of the final pose of the arm after rolling out the learned BC reaching policy.

The **********8 (Schulman et al., 2017)

6.4. Behavioral Cloning on the Real Robot

While we were not able to train the full BC+RL algorithm on real robots, we were able to train the pure BC segment of the algorithm on a restricted task. In the real world task, we train a BC policy to attempt a grasp of the corner of a folded towel (see Fig. 7). To do this, we gather 30 human demonstrations in approximately 20 minutes of human time which corresponds to about 30 seconds per episode with data recorded at $30hz$. The policy can successfully reach the towel; however, grasping proves difficult as the data imbalance is high. While grasping the towel and unfolding the towel is the full task proposed, learning subtasks (like reaching) quickly with minimal real robot time (which would require human resets for many tasks) is extremely beneficial and would greatly simplify the exploration problem.

Additional difficulties exist with applying a BC policy to real robots: the timing. The policy was trained to emulate actions at $30hz$ so when rolling out the policy, care must be taken to ensure that the rollout occurs at 20hz as well. We also note that the method of control on the real robot is not entirely accurate. The target locations returned by the IK computations do not directly account for the deltas in position and orientations at the end effector. For this reason, we moved from using the human actions in delta end effector space to using deltas from the robot's proprioception of its end effector pose. With a perfect controller, there would be no difference between these two inputs; however, we find that empirically working with deltas computed from proprioception attains better results.

7. Conclusion

For long horizon tasks with sparse rewards, training RL algorithms to achieve success is difficult at best and often times impossible. This is due to the exploration problem. Furthermore, naive behavioral cloning will often times fail to attain the performance desired on randomized environments, and especially so when demonstrations are not optimal. We show that it is possible to address the exploration problem of RL and the lack of robustness of BC by combining them both. Such an algorithm leverages the strengths of both and uses each to address the deficiencies of the other. We show that one possible method of combining these two algorithms is a simple convex combination of the losses with a schedule on the weighting parameter. This schedule allows to get a "behavioral cloning initizaliation" early in training that allows for efficient exploration of the the RL agent later in training when the weighting puts more weight toward the RL loss. We show that this solution is able to achieve high returns on the environment as well as decreased wall-clock time to convergence which both suggest that single agent algorithms may be able to solve these tasks when using this approach.

The approach for behavioral cloning taken in this work is largely domain specific for the tasks being examined, but the algorithm for combining BC and and RL is general: any BC algorithm and any RL algorithm can be combined in this way. This approach is a first step toward algorithms that allow RL agents to ingest large amounts of demonstrations and be able to learn from demonstrations that are sub-optimal.

8. Future Work

Possible areas for future work include introducing correlated noise into rollouts from the optimal learned policy to evaluate the potential for this algorithm to learn from multi-modal data. For example, have different means for the noise, or draw from beta distributions with difference parameters to represent different users. This would better approximate the data collected from systems like RoboTurk. Additional work should be done to improve the IK control

in simulation to allow for the training of optimal policies using deltas in end effector pose. Specifically, addressing the memory leak issue as well as improving the responsiveness of the control. Additionally, improvements should be made to the real robot pipeline to enable training of RL safely on the robots and to allow more accurate movement in space using IK or impedance control to allow for manipulation at the end effector.

References

Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., and Krishnan, D. Unsupervised pixel-level domain adaptation with generative adversarial networks. *CoRR*, abs/1612.05424, 2016. URL http://arxiv.org/abs/1612.05424.

Fan, L., Zhu, Y., Zhu, J., Liu, Z., Zeng, O., Gupta, A., Creus-Costa, J., Savarese, S., and Fei-Fei, L. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on Robot Learning*, 2018.

Ho, J. and Ermon, S. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016. URL http://arxiv.org/abs/1606.03476.

Mandlekar, A., Zhu, Y., Garg, A., Booher, J., Spero, M., Tung, A., Gao, J., Emmons, J., Gupta, A., Orbay, E., Savarese, S., and Fei-Fei, L. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

Riedmiller, M. A., Hafner, R., Lampe, T., Neunert, M., Degrave, J., de Wiele, T. V., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing - solving sparse reward tasks from scratch. *CoRR*, abs/1802.10567, 2018. URL http://arxiv.org/abs/1802.10567.

Salimans, T. and Chen, R. Learning montezuma's revenge from a single demonstration. *CoRR*, abs/1812.03381, 2018. URL http://arxiv.org/abs/1812.03381.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Zhang, T., McCarthy, Z., Jow, O., Lee, D., Goldberg, K., and Abbeel, P. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *CoRR*, abs/1710.04615, 2017. URL http://arxiv.org/abs/1710.04615.

Zhu, Y., Wang, Z., Merel, J., Rusu, A. A., Erez, T., Cabi, S., Tunyasuvunakool, S., Kramár, J., Hadsell, R., de Freitas, N., and Heess, N. Reinforcement and imitation learning for diverse visuomotor skills. *CoRR*, abs/1802.09564, 2018. URL http://arxiv.org/abs/1802.09564.