Improving Reinforcement Learning Exploration for Manipulation with Intrinsic Rewards

Eric Chen
CSAIL, MIT
ericrc@mit.edu

Isaac Perper
CSAIL, MIT
iperper@mit.edu

Michael Hiebert
CSAIL, MIT
mhiebert@mit.edu

***Abstract*—Real world manipulation using reinforcement learning is challenging and data inefficient. Because of this, traditional reinforcement learning approaches tend to fail beyond only the most simple manipulation tasks due to an inability to successfully explore their environment and efficiently sample enough data. In this project, we seek to understand the effectiveness of two exploration methods (ICM and Disagreement), in the context of solving a manipulation task. We implement both approaches, and probe the limitations of both methods as we transform our learning environment from an ideal-state-simulation to one with more resemblance to the real world.**

***Index Terms*—Robotics Manipulation, Exploration, Reinforcement Learning**

I. Introduction

In recent years, reinforcement learning (RL) approaches have seen some success in solving manipulation tasks [1], [2]. However, one fundamental issue limiting the use of reinforcement learning on manipulation tasks is the sparsity of the extrinsic reward function. The fundamental idea behind RL is that a feedback signal or ‘reward‘ from the environment can direct the agent to learn a policy through a trial and error process. The easiest way to define a reward is to reward success, and give zero reward for all other outcomes. This sparse reward setting is difficult because the agent does not receive meaningful feedback for its actions very often, making such methods quite sample inefficient [1]. By including various rewards for other intermediate outcomes, the negative effects of this problem can be lessened. With a dense reward signal, RL has been shown to work well in simulated environments like video games [3], [4]. Unfortunately, it is difficult to come up with dense reward functions that generalize across objectives for real world manipulation tasks [2].

Some approaches attempt to address this issue with a dense intrinsic reward that supplements the sparse extrinsic reward by encouraging the agent to explore more novel states. With an intrinsic reward, the agents themselves compute a reward based on their state and understanding of the environment. This allows agents to learn basic skills that allow them to better achieve the sparse extrinsic reward they are unlikely to encounter otherwise. One simple method is maximizing visitation count [5], but this suffers from the fact that not all states are useful for learning. A lot of work focuses on rewarding movement into 'learnable' states [6], [7], but what is actually most useful to learning is challenging to determine.

In this paper, we explore a novelty-based curiosity method [3], which uses a forward model to predict the next state, and formulates the intrinsic reward as the mean squared error between the prediction and the ground truth. The intuition is that if the agent is not able to predict the consequences of it's actions, it should be interested in exploring those actions in the future. If it is capable of making good future predictions, then it has adequately learned the dynamics in that part of the space and should move on to more interesting things.

With sparse extrinsic rewards, manipulation tasks stand to benefit from this curiosity framework. In addition, ICM [3] is able to handle stochasticity in the environment. In real-world manipulation tasks, environments are filled with many dynamics that are inherently unpredictable - such as leaves rustling in a tree. On its own, a prediction based curiosity method such as ICM would never learn to perfectly predict the forward dynamics of the leaves, and would always be interested in them even if it cannot affect the leaves with it's actions. ICM solves this by using an inverse model, that predicts the action given a state at time t and the next state at time $t+1$. Because the leaves are not affected by any of the actions the robot takes, they are irrelevant to the action prediction task. As such, the inverse model learns to encode observations in a latent space where only action-relevant information about the dynamics of the world exist. By encoding observations into this feature space before passing them into the forward model, ICM is capable of ignoring distracting environmental noise.

Handling environment based noise is great, but real-world manipulation also encounters action based stochasticity. Servo noise, gears imperfections, and contact forces can all lead to slight variations in the forward dynamics given a specific action. These sources of noise are distracting to the forward model, and because they affect the action prediction task they are not dropped in the latent space trained by the inverse model. In a more recent paper, a curiosity algorithm called Disagreement [8] extends ICM and makes it robust to action based noise. Disagreement trains multiple forward models of the environment, and takes the variance across the predictions as the intrinsic reward. When an action is non-deterministic and leads to a distribution over future states, all of the forward models converge to the mean prediction and the variance goes to zero. Disagreement is supposedly capable of operating on a real robot, and would be capable of working in a manipulation

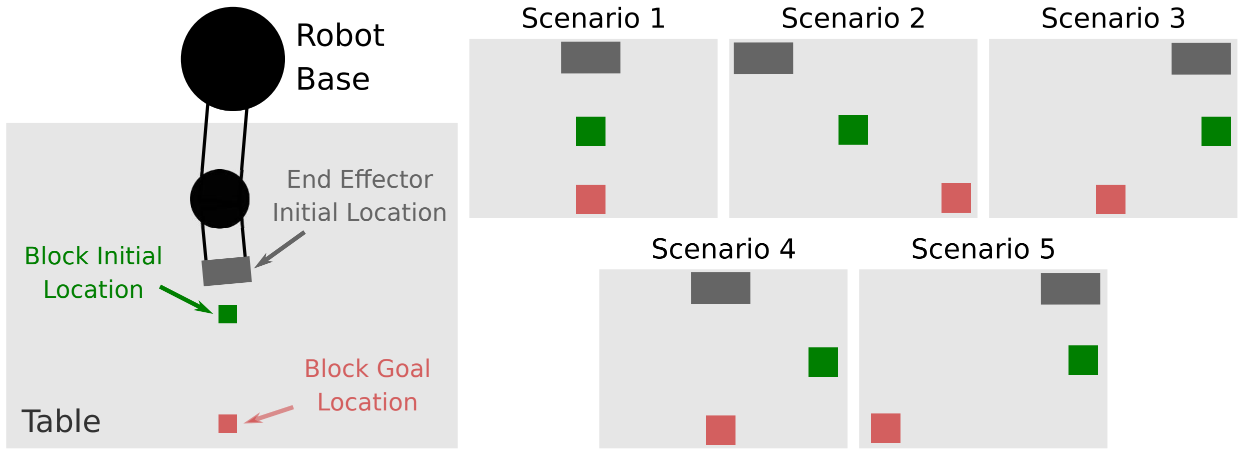


Fig. 1: The experimental scenarios used throughout this report. Each experiment has a different initial block location, initial end effector location, and a block goal location. Scenarios are ordered by relative difficulty, with Scenario 1 being the easiest.

setting with both environment and action based noise.

Our paper focuses on testing the ICM and Disagreement approaches on a push manipulation task in an OpenAI Gym environment. We compare performance of PPO augmented with ICM, with standard PPO using ϵ-greedy exploration. In an effort to study how well these methods might scale to real manipulation tasks, we study more complex action spaces, and we also analyze how ICM and Disagreement compare when faced with noisy actuation.

II. Experimental Setup

A. Simulation

We used the Mujoco Fetch robot environment [9] in OpenAI Gym [10] as our simulator. This environment has a 7-DoF Fetch Mobile Manipulator fixed next to a tabletop containing a black cube that the arm can interact with. We focused on a simple push task to simplify the learning in scope of the class project, and the gripper remained in a closed state to simplify the action space. The Mujoco backend simulation operates at a much smaller delta timestep than the gym environment, so it accurately captures contact dynamics for the purposes of this project. We explored a variety of camera placements for the task including fixed downward facing and wrist mounted cameras, but decided on the camera view shown in Fig. 2. This view provided relatively few obstructions of our target block object while still discerning arm and block spatial displacements well. Additionally, this view might be a fairly realistic way of mounting a camera given a fixed workstation. For future work, it might be interesting to see how these methods perform given observations gathered from cameras mounted on the robot itself.

B. Scenarios

The five different training scenarios we chose to study (Fig. 1) are meant to provide varying levels of task difficulty, giving us the opportunity to perform a more comprehensive comparison between our PPO baseline and it's ICM augmented counterpart. The only differences between scenarios are the initial locations of the block and gripper, and the goal location of the block. The extrinsic reward function used is sparse - the agent is given 0 reward everywhere, unless the block is within a small threshold of the goal location. If the object is within this threshold, the agent is given a reward of 1. Each rollout lasts 50 timesteps.

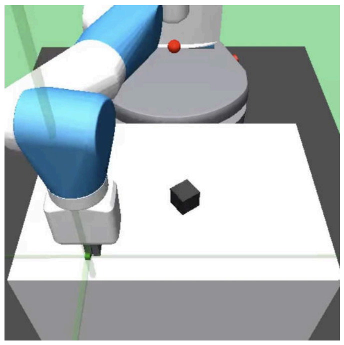
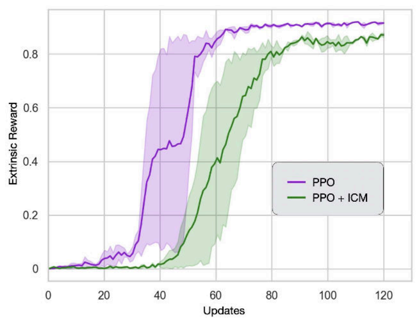


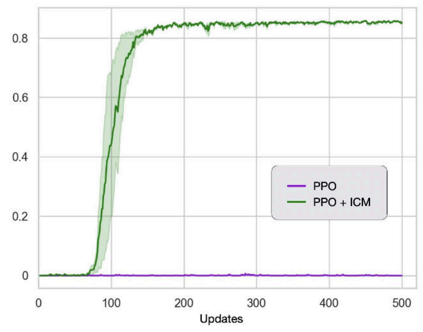
Fig. 2: The camera view used for experiments. This angle gives a clear view of the black block and the arm, and changes in the blocks position on the table are clearly noticeable. All observations used in our experiments are RGB images from this point of view.

C. Training

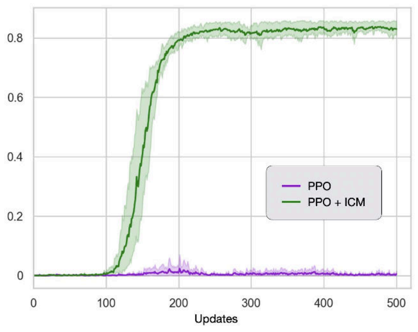
We implemented both ICM and Disagreement in PyTorch for training. PPO [11] was used as the base learning algorithm, and we used the rlpyt open-source codebase [12] as a basis for our sampling infrastructure. We ran 128 environment instances in parallel, and collected 50 step rollouts on each. Training was conducted using a batch size of 200 samples per batch. Observations consisted of (500, 500, 3) RGB images, which were downsampled to (84, 84, 1) greyscale images using bilinear interpolation in order to fit in memory. Actions consisted of discrete integers that mapped to specific torque commands. We explored a variety of action spaces in our experiments. We used a learning rate of $1e^{-4}$ for the policy



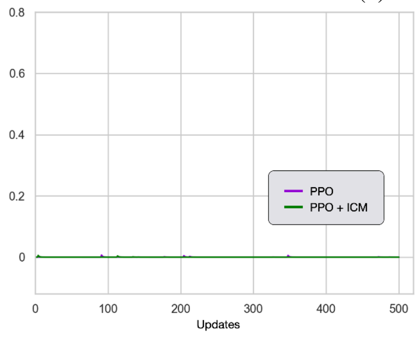
(a) Scenario 1



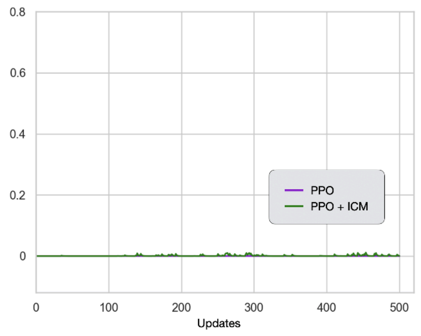
(b) Scenario 2



(c) Scenario 3



(d) Scenario 4



(e) Scenario 5

Fig. 3: Performance of PPO and ICM augmented PPO on all five block pushing tasks. Each curve represents 5 random seeds, with standard deviation. These plots make it clear that certain tasks are achievable with ICM augmented PPO.

network, the value network, and the curiosity networks. As suggested in a followup paper to ICM [13], we normalized intrinsic rewards and advantages to stabilize training. Batch normalization was also applied to both the forward and inverse curiosity models. Lastly, we use an entropy term coefficient of 0.001 for both PPO, ICM and Disagreement in order to manage the exploitation-exploration trade-off and encourage the agent to discover new sources of extrinsic or intrinsic rewards.

III. Experiments and Results

A. Baseline

To begin, we simplified the continuous action space of the push environment into a discrete action space where the end effector is constrained to 2D movement on the table's surface. This base action space consisted of 8 unit actions in each direction, and an additional null action where the arm doesn't move. We tested both PPO (extrinsic reward only) and ICM augmented PPO (extrinsic + intrinsic reward) on each of the 5 scenarios. Fig. 3 shows these results.

In Scenario 1, the robot is able to push the block to the goal position using both PPO and ICM. This is not surprising, as the starting location and the goal location of the block and gripper are relatively close to each other. Interestingly, PPO learns faster than ICM. This could be because PPO latches onto the extrinsic reward immediately, while ICM gets ”distracted” exploring interesting parts of the space using it's intrinsic reward before focusing on the extrinsic reward signal.

In Scenarios 2 and 3, PPO is unable to reach the goal, while ICM is able to. These scenarios are hard in two separate ways. In Scenario 2, the gripper has to travel across the entire diagonal of the table and push the block further than it has to in Scenario 1. This makes Scenario 2 difficult to solve using epsilon-greedy exploration, where the agent needs to get lucky in order to stumble into the sparse extrinsic reward. ICM however, learns to interact with the block out of interest, and in doing so is able to run into the sparse extrinsic reward signal more often.

In Scenario 3 the gripper can easily knock the block off the table. In fact, with the block starting at the edge, there are few approaches to contact with the block that will lead to a stable policy. Knocking the block off the table represents a potential local maxima in the intrinsic reward space, and the agent must learn to push the block inwards towards the goal in order to succeed. PPO is not able to solve this task, whereas ICM is.

Lastly, we found that neither approach could reach the goal in Scenarios 4 and 5. Both of these show a combination of long travel distance and near-edge block initialization, which combines the previous issues. In Scenario 4, we find that ICM gets stuck pushing the block off the table to generate prediction reward, and doesn't discover the greater source of intrinsic reward to be had if it goes around and pushes the block inwards. This is a demonstration of a key flaw with ICM, that might easily make it difficult to scale to more complex manipulation tasks. Specifically, ICM still needs an entropy term to allow the agent to explore outside of its current policy and discover new sources of intrinsic reward. Even with intrinsic rewards, you can get stuck in local optima that will hinder progress and stop you from learning more interesting behaviors. In Scenario 5, the distance the block has to travel in order to get to the goal is the greatest out of all the scenarios. In this situation, both PPO and ICM are not able to solve the task. Because ICM is able to solve tasks such as Scenario 3 and 2, the inability to solve Scenario 5 is most likely a reflection of the short term nature of ICM. Because rewards are formulated based on a 1-step prediction, ICM might struggle to learn more complex manipulation behaviors that require longer term reasoning.

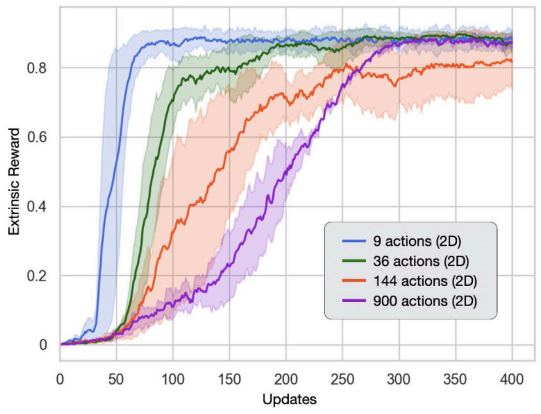
B. *Discrete Action Space*

While a discrete action space is interesting and allowed us to compare ICM and PPO on a variety of push tasks, real world manipulation tasks would most likely require a continuous action space. In order to study performance as we move towards a continuous space, we further discretized the action space and increased the number of available actions. We did this by both increasing the granularity of movement by reducing our grid size, and increasing the number of directions for possible movement. In this set of experiments, we also allow z-axis movement and no longer constrain the end effector to the plane of the table in order to achieve a "3" action space. The environment wrappers and their corresponding action space sizes can be seen in Table I.

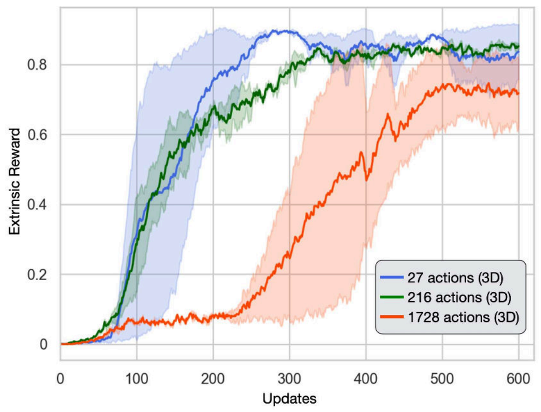
Wrapper	Grid Tile Size	Action Space Size
Base2D	0.03^2	9
Base3D	0.03^3	27
Micro2D	0.015^2	36
Micro3D	0.015^3	216
Nano2D	0.0075^2	144
Nano3D	0.0075^3	1728
Pico2D	0.001^2	900
Pico3D	0.001^3	—

TABLE I: The different action space discretizations used to test the impacts of a more complex action space. These were implemented using OpenAI gym environment wrappers, and multiple levels of spacing and dimensions were chosen. The tile sizes given are in the gym's unit coordinates.

Fig. 4 depicts the results of changing the discretization of the action space according to Table I. These results were run on Scenario 1 with an extrinsic reward for both PPO and ICM. In the different 2D action spaces, the finer grid spacing only slows down learning, rather than causing it to fail completely. The 3D action space also converges to a decent solution even with 1728 discrete actions per step. In both scenarios, there is a clear separation between action spaces, with coarser, larger action spaces learning slower. It's important to note that we did not have time to test discretization on more challenging scenarios, which we suspect may not work with higher grain action spaces. Still, the results show that ICM is surprisingly robust to increasingly large action spaces. However, our results indicate it might fail when presented with more challenging



(a) 2D Action Discretizations



(b) 3D Action Discretizations

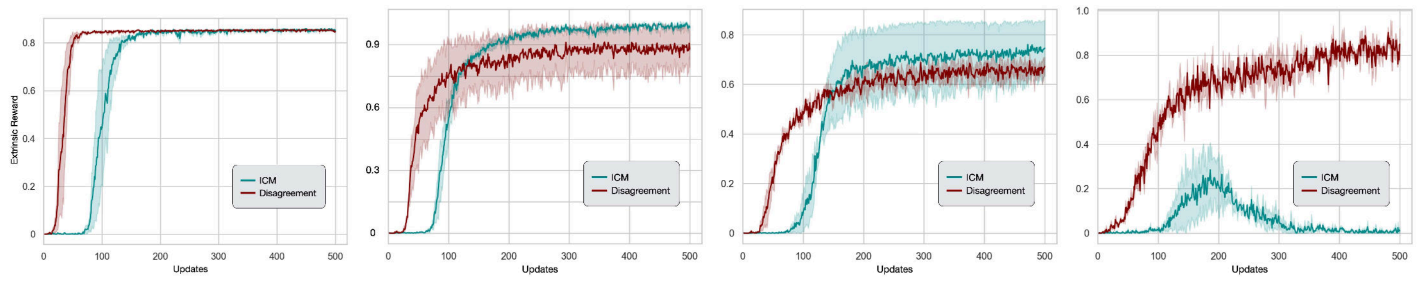
Fig. 4: Performance of ICM with different action spaces on the Scenario 1 task, with extrinsic rewards. Each curve represents 5 random seeds, with the standard deviation. The last of the

tasks, and a continuous action space. More experiments would be needed in order to determine the failure threshold.

C. *Action Noise*

Our initial experiments were simplified with a deterministic simulation, such that the robot would always end up at the exact same next state given an action. This provides a baseline on whether or not the benefits of ICM are noticeable in a no-noise case. Since real-world robots have to confront sources of action based noise (noisy servos, mechanical imperfections, etc.) we wanted to test how robust ICM is to action based noise. We also wanted to test if Disagreement offers significant performance improvements in this setting when compared to ICM. We compare no (0%), low (2%), medium (10%), and high (30%) noise levels on the forces applied in the x and y directions during each step. These were trained in Scenario 1, but it would be interesting to train these on the more challenging scenarios.

Fig. 5 shows the impacts of varying levels of noise on the actions on the performance of ICM and Disagreement. On the easier experiments with a 0% and 2% noise, both ICM and Disagreement solve the task. Surprisingly, ICM also suc-



(a) No noise (b) 2% noise (c) 10% noise (d) 30% noise

Fig. 5: Performance of ICM and Disagreement in Scenario 2 when noise is applied to the actions. Each curve represents 5 random seeds, with standard deviation. For the y-axis, 1.0 is the maximum achievable extrinsic reward. These plots show that both ICM and Disagreement can cope with low noise, but only Disagreement works at high levels of noise.

ceeds with 10% noise. At 30% noise, Disagreement performs significantly better, and ICM is not able to learn the correct policy. It's quite amazing that Disagreement performs at this level of noise, and shows promise for the model potentially handling larger perturbations. While we were interested in smaller sources of noise such as servo noise, it's interesting to see that these methods can handle fairly large amounts of noise in the actions. Testing these on a real robot might be an interesting next step.

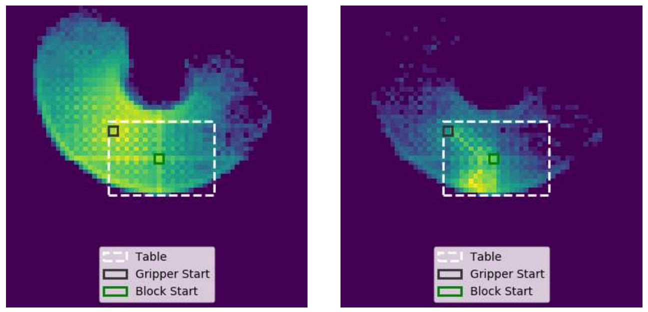
D. *Exploration*

To directly picture the exploration strategy employed by ICM, Fig. 6 illustrates a heatmap of how often the gripper visited each location in the environment with random exploration versus ICM exploration. As expected, random exploration visits a wide range of positions in the environment. In comparison, ICM clearly focuses its attention around the table and the location of the block. This is a good indication that ICM is performing as it is designed. Actions with the block are inherently more interesting, given that they change more features of the environment than the robot arm moving without any interactions.

Another way to look at this is through the lense of sample efficiency. When it comes to learning manipulation and planning on real-robots, sample efficiency is essential. By providing a smarter way of exploring space, every step can provide maximal information gain, and thus, be trained faster. As shown in the heatmaps, ICM would allow a robot interact with the block directly more often, guiding the learning in a more efficient manner.

IV. Future Work

This project highlighted some interesting research directions for methods like ICM in the context of robot manipulation. While our results showed that ICM was capable of completing tasks that PPO was not able to, we did highlight a key limitation surrounding the entropy term. As such, it might be interesting to see if these methods could be improved for manipulation tasks by encoding observations in a task-relevant latent space, where the state encodes information about objects of interests in the environment. While this does go against one of the main appeals of intrinsic motivation by constraining the search space and baking in additional information, it might allow us to perform better on practical tasks such as grasping or pick and place.



(a) Random Exploration Heatmap (b) ICM Heatmap

Fig. 6: The visitation heatmaps of random exploration and icm exploration, where the brightness corresponds to the log of the number of times the gripper visited that location during training. The table is outlined, along with where the gripper and block are initialized. The arc shape comes from the dynamics of the arm rotating on a base. The random exploration searches the whole environment, even though many parts of the environment don't contain interesting features. In contrast, ICM focuses near the table and around the block.

Grasping in particular would be an interesting next step. In our results, we showed that training failed for 2 out of 3 scenarios where the block was near the edge. This may be due to there being a small set of possible actions near the block that succeed rather than push it over the edge. Similarly, grasping is a task where there are a limited number of ways to succeed in grasping an object, and the unrefined PPO model may struggle in these situations.

Additionally, working towards a continuous action space is an important step towards deploying these models in a real manipulation setting. While we did show that a larger number of actions adversely affects performance in one of the simpler scenarios, it would be interesting to test with a variety of action spaces on all of the scenarios in order to see if performance breaks down more drastically in the more challenging experimental scenarios. To move to a fully continuous action space, work would most likely have to be done to improve the inverse model.

V. Conclusion

While reinforcement learning has pushed the boundaries on many tasks like gaming, end-to-end learning of real-world manipulation tasks remains a challenge. Even if end-to-end isn't the right approach, modular manipulation components such as task planning still require a way of deciding the best action given many environmental agents, noise, static objects, and system dynamics. Efficiently exploring the environment is one way to tackle this high-dimension state space, and this project shows how intrinsic rewards can accomplish this. We showed the ICM can accomplish a larger and more complex set of tasks, and that Disagreement can handle the actuation noise that a real robot might encounter when interacting with the world.

Moreover, even in absence of an external reward or "goal", an intrinsic reward was able to explore the environment efficiently, interacting with the interesting table and block features. This supports the notion that intrinsics can be used for learning core ideas like how to move around in space. Although we didn't have time, these intrinsics could be used as a pre-training step if we were to try more complex tasks like grasping. With manipulation, determining what can be learned and what we have tools for already is an important task, and better, more efficient exploration can get us closer to what works and what can be improved.

Code

All code along with sample videos and relevant documentation can be found here: https://github.com/echen9898/curiosity_baselines/tree/881_project

Acknowledgment

We'd like to thank Prof. Tedrake and the 6.881 staff for giving us useful advice throughout the semester and helping us work on our final project.

References

[1] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[2] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

[3] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven Exploration by Self-supervised Prediction," *CoRR*, vol. abs/1705.05363, 2017, _eprint: 1705.05363. [Online]. Available: http://arxiv.org/abs/1705.05363

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv:1312.5602 [cs]*, Dec. 2013, arXiv: 1312.5602. [Online]. Available: http://arxiv.org/abs/1312.5602

[5] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying Count-Based Exploration and Intrinsic Motivation," *arXiv:1606.01868 [cs, stat]*, Nov. 2016, arXiv: 1606.01868. [Online]. Available: http://arxiv.org/abs/1606.01868

[6] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," 1991.

[7] B. T. Lopez, J.-J. E. Slotine, and J. P. How, "Dynamic Tube MPC for Nonlinear Systems," *arXiv:1907.06553 [cs, eess]*, Jul. 2019, arXiv: 1907.06553. [Online]. Available: http://arxiv.org/abs/1907.06553

[8] D. Pathak, D. Gandhi, and A. Gupta, "Self-Supervised Exploration via Disagreement," 2019, _eprint: 1906.04161.

[9] Roboti, "Mujoco." [Online]. Available: http://www.mujoco.org/index.html

[10] OpenAI, *A toolkit for developing and comparing reinforcement learning algorithms*, publication Title: Gym. [Online]. Available: https://gym.openai.com/envs/FetchPush-v0/

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

[12] A. Stooke and P. Abbeel, "rlpyt: A research code base for deep reinforcement learning in pytorch," *arXiv preprint arXiv:1909.01500*, 2019.

[13] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," *arXiv preprint arXiv:1808.04355*, 2018.