Intrinsic Motivation for Robotics Grasping

Antony Zhao
ayzhao7761@berkeley.edu

Junhua (Michael) Ma
junhua@berkeley.edu

***Abstract*—In this project, we explore whether intrinsic motivation based on prediction error can enable a robot to learn grasping without relying on explicit rewards. We trained a Sawyer robot in simulation using PPO with and without Random Network Distillation (RND), comparing performance in dense and sparse reward settings. Although RND did not improve outcomes in our simple grasping tasks, it demonstrated interesting exploratory behaviors, which shows promise when used for more complex robotics tasks. Additionally, we deployed our trained RND agent to test in the real world to confirm its real-world applicability.**

I. Introduction

Reinforcement learning (RL) has emerged as a powerful framework for enabling agents to acquire complex behaviors through interaction with their environment. However, in scenarios where external rewards are sparse or difficult to define, agents often struggle with effective exploration, hindering learning progress. To address this, intrinsic motivation mechanisms where agents generate internal rewards based on factors like prediction error have been proposed to encourage exploration by quantifying novelty or surprise in the environment [1]. Methods such as Random Network Distillation (RND) is a effective implementation of this approach for improved learning performance [2].

Robotic grasping is a fundamental yet challenging problem in the field of robotics, involving complex processes generally including object detection, pose estimation, motion planning, control, all of which must be executed with high precision to manipulate objects of varying properties. Despite significant advancements, especially with rising popularity of using RL-based methods to learn grasping, achieving human-level dexterity and adaptability in robotic grasping remains a challenging problem.

Our project explores the application of RL and intrinsic motivation, specifically with intrinsic rewards based on prediction error following the RND implementation, to facilitate learning in robotic grasping through human-like self-driven exploration with limited or no explicitly defined extrinsic rewards. We implemented and evaluated this approach in a simulated environment using the Robosuite framework [3] with a Sawyer robot and deployed it to a real-world environment to test the practicality of our methods. By comparing performance across dense, sparse, and purely intrinsic reward settings, we assess the effectiveness of intrinsic motivation and discuss findings and potential future directions. Overall, our work contributes to the broader goal of enabling more autonomous, generalizable, and practical robot learning, with robotic grasping as a starting point.

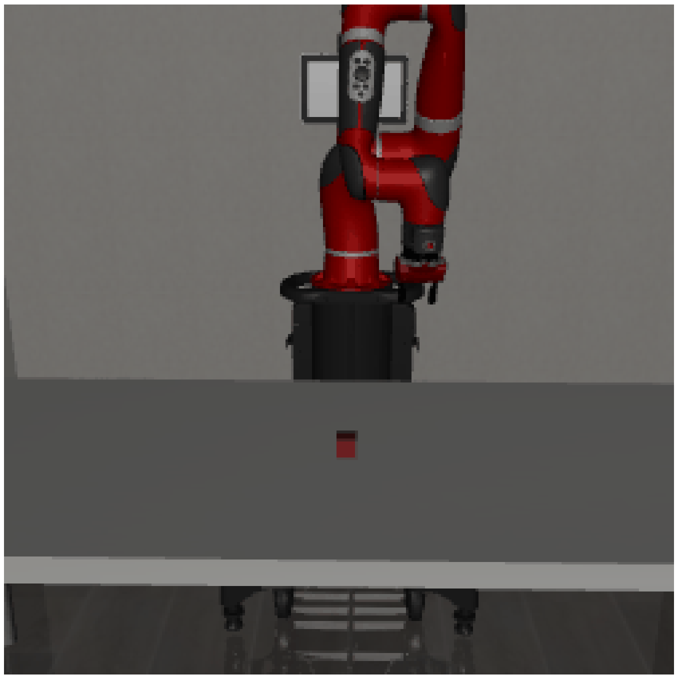
Fig. 1. Simulation Environment.

II. Related Work

A. Reinforcement Learning

Reinforcement learning (RL) has been widely applied to robotic control tasks, including locomotion, navigation, and manipulation. RL enables agents to learn policies directly from interaction, which doesn't require the extensive datasets required by other machine learning methods like imitation learning [4], and commonly results in greater adaptability than classical control methods. Most notably, RL training is inspired by the human learning process and can lead to learning intelligent, human-like behaviors, which are very desirable in robots. However, RL methods generally require simulations as training environments and encounter challenges with real-world deployment due to gaps between simulation and reality. The performance of RL training also varies considerably based on many possible hyperparameter configurations and requires a well-defined reward function, which further limits its practicality to real-world robotic tasks. In the context of robotics grasping, RL methods overall demonstrate successful results even for more complicated tasks like grasping in cluttered environments, where RL agents can learn to interact with the environment to achieve successful grasp even in the presence of obstacles [5]. Further works explore more complex network architectures to also optimize the efficiency of grasping and environment interaction to eliminate unnecessary movements [6]. However, prior works require complex workflow and specially crafted metrics to generate well defined reward functions, which limits the generalization and scalability of the overall learning setup.

B. Intrinsic Motivation

Intrinsic motivation has been proposed to enhance RL learning, particularly in environments where external rewards are sparse or difficult to design. Methods such as curiosity-driven exploration and prediction error-based bonuses have demonstrated strong performance in complex exploration tasks, especially in video game benchmarks with sparse rewards [1] [2]. In the field of robotics, intrinsic motivation has been applied to improve learning in tasks involving locomotion, navigation, and manipulation [7], helping agents explore more effectively without dense supervision. However, the application of intrinsic motivation remains relatively limited in scope, and to our knowledge, no prior work has directly investigated its use for robotic grasping tasks—a domain where reward design can be especially challenging and sparse feedback is common.

III. Methods

A. Proximal Policy Optimization (PPO)

We utilize PPO as our RL algorithm across all experiments. It is widely used in robotics and is commonly used in ALE environments, where much of the work for intrinsic motivation is done. We use a fairly standard PPO algorithm using generalized advantage estimation A and policy gradient clipping, as shown by the equation below.

$$L(s,a,\theta,\theta_{old}) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}A, clip\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}, 1-\varepsilon, 1+\varepsilon\right)A\right)$$

Actions are sampled from a normal distribution with a state-independent log standard deviation, and the policy network computes a mean for this distribution. These are then passed into a tanh function before being passed into the environment to normalize the actions into a reasonable range for robotic control.

B. Random Network Distillation

Random Network Distillation (RND) utilizes the fact that model error is high when there are few samples for the model to train on. The original paper focuses on the Montezuma's Revenge environment from ALE, where changing rooms create significant differences in the image observations and help direct the agent where it would otherwise be difficult to explore the environment.

This works by randomly initializing two feature models $f_{target}, f_{predictor}$. The target model is frozen for the entire process. We then compute the mean squared error loss on the outputs of the two models on states. We both train the predictor model to minimize it and use it as our intrinsic reward. As another implementation detail, we keep a running standard deviation, which we use to normalize the intrinsic rewards to a reasonable level.

$$\|f_{target}(s_{t+1}) - f_{predictor}(s_{t+1})\|$$

To modify PPO to fit RND, the value network outputs two values, one for the extrinsic value V_E and one for the intrinsic value V_I. These are combined when computing the advantages for the policy gradient, specifically, we do not use episode terminations while computing the V_I, and we normalize the A_I, A_E separately before adding them together to get $A = A_I + A_E$.

C. Neural Network Architecture

For our observations, we have image data from a simulated camera as well as various proprioceptive observations for the robot itself. However, we didn't have access to some of these on the robot in the lab, so the final robot observation is a 37-dimensional vector containing information about the joints, end effector, and gripper. We also get a 100×100 image that we grayscale.

We also use two heads for the model: a convolutional head that processes the images and a linear layer that processes the robot observations. We then concatenate these observations together.

We also utilize observation stacking for both the policy and value networks in our RL algorithm, which gives us a 4-channel 100×100 image and a 148-dimensional vector, which are passed into the model. For the feature models in RND, we only take the latest frame and robot state, as this was the original way it was implemented for Atari. We also normalize/scale the observations to a reasonable range for the models. For the policy/value networks we scale the image observations such that the pixel values are from $[0, 1]$, and multiply the robot state by 10 to get them to a more reasonable range. For the feature networks in RND, we keep track of a running mean and standard deviation, which are used to normalize the image observations, and we keep the robot state observations the same.

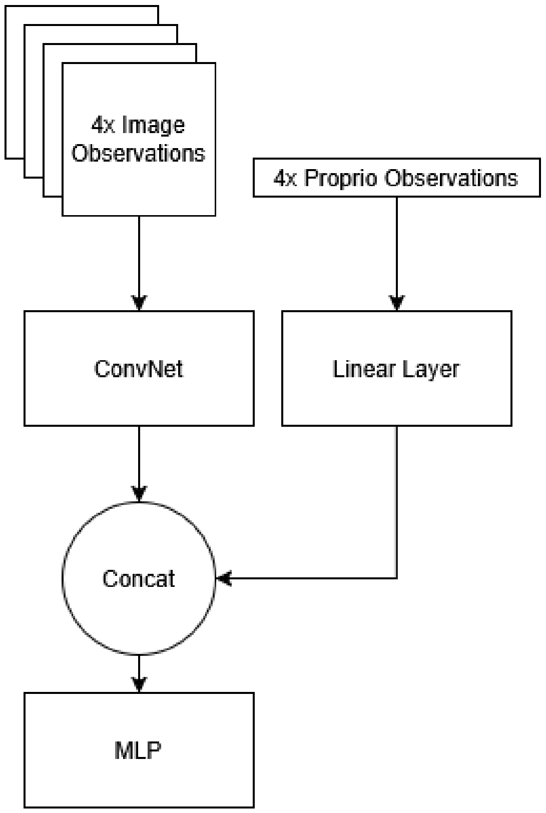

Fig. 2. Core of RL models

IV. Results

A. Simulation

We ran several experiments in simulation to analyze the effects of intrinsic rewards on the agents' learning speed. Unfortunately, intrinsic rewards seemed to be more of a distraction than a helpful guide to our models, as they performed worse in both the dense rewards (which provide rewards for actions such as reaching, grasping, and picking up) and the sparse rewards (which only provide rewards upon lifting).

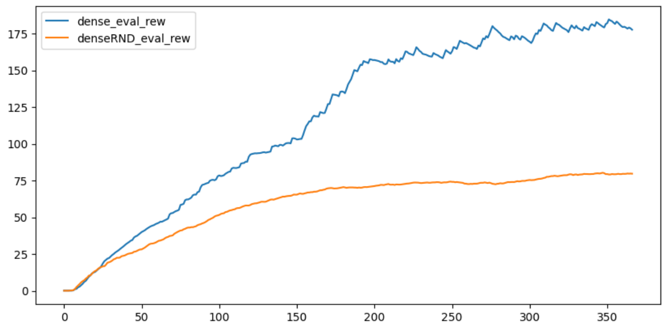

Fig. 3. Training results with dense rewards

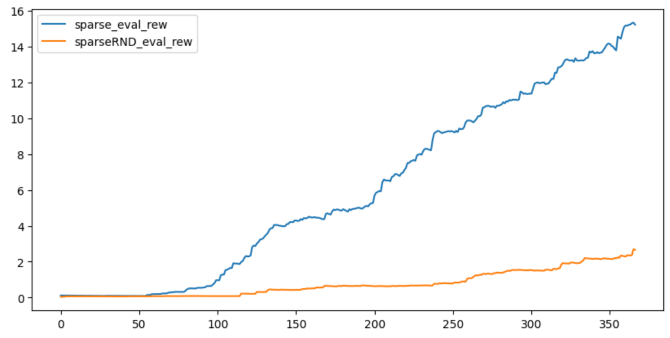

Fig. 4. Training results with sparse rewards

We kept the RND module training in the background for all of these experiments, even if we didn't use the intrinsic reward to train the agents themselves. This did show an interesting result, in that the agents that were solving the regular task were also increasing the amount of intrinsic reward to the agents, which was an encouraging result. As such, we also trained models only to utilize the intrinsic reward without any extrinsic reward, to analyze what the agents would do.

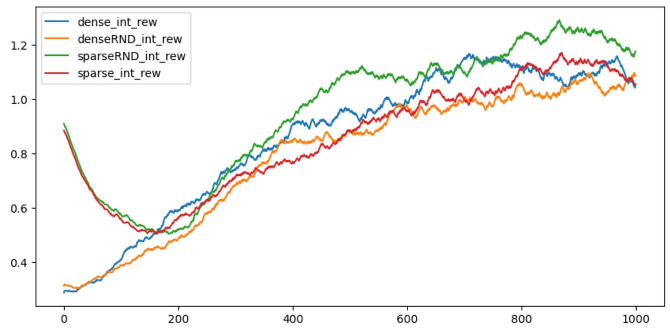

Fig. 5. Intrinsic reward over time, all models tended to increase this.

The resulting policies are shown in the appendix website, but generally, we found that the agent learned to keep the arm inside the camera workspace and would often hit the block itself, which led to higher extrinsic rewards without being explicitly trained on it. We also compared how these agents behaved compared to a completely random policy, which we noticed tended to extend the arm completely and get stuck, often outside of the workspace/camera. Below is also a comparison of the pure RND agent to the sparse reward agents (both RND and no intrinsic). While this isn't a fair comparison due to a variety of changes to hyperparameters/system dynamics to allow us to do sim-to-real on these policies, the fact that it was performing better than our sparse + RND did show that there was certainly room for improvement in the performance as well.

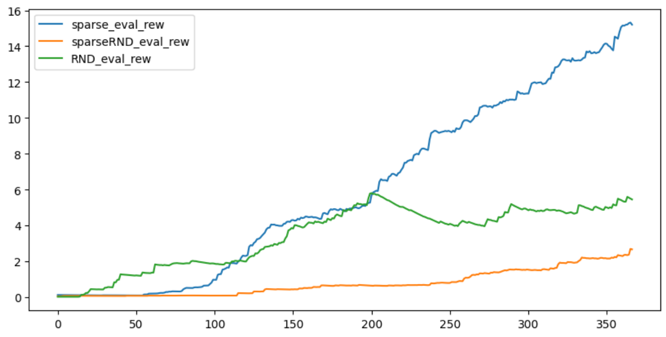

Fig. 6. More training comparisons

B. Hardware

To bridge the sim-to-real gap, we utilized a built-in observation randomization wrapper and modified the environment to match the level of the table with the one in the lab. We specifically wanted to see the dense RND policy and a pure RND policy. We use a Logitech C920 to get images at 30 fps, crop/scale/grayscale to mimic our simulation observations, and extract the robot state using ROS. Our simulation ran at 10 Hz and matched that frequency in ROS. Many of the shown training curves were also used operational space control in simulation, but we were using joint velocity control so we had to retrain the dense RND model, and the pure RND model was trained using the joint velocity control as well.

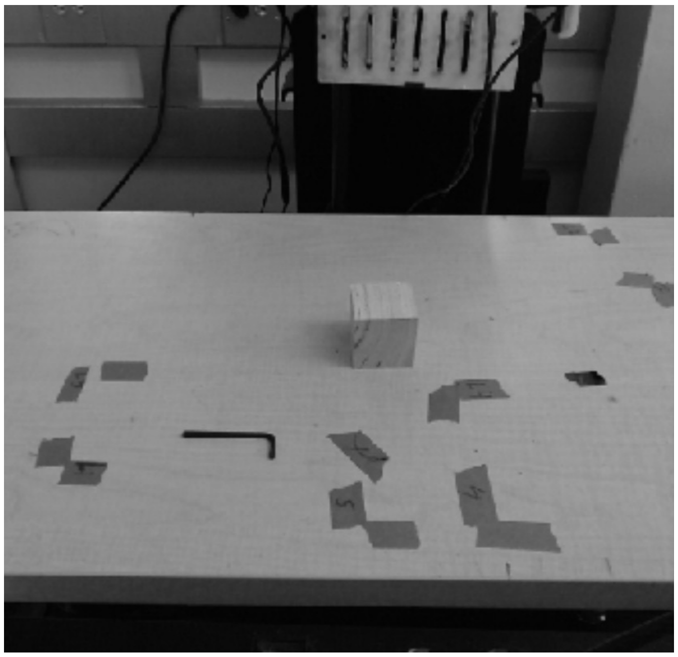

Fig. 7. Grayscaled images from the webcam

Unsurprisingly, given the learning curves, our dense RND model, while it could reach for the block, tended to be a bit unstable as the policy in simulation was likely exploiting the intrinsic reward rather than focusing on the task itself. Interestingly, our RND policy acted similarly between simulation and reality by moving around in the camera space, and would go near the block. We specifically chose a policy that wasn't tending towards the camera, as there was likely no physical camera in the simulation, which wouldn't match reality. We also compared this to passing randomly sampled actions to the robot by sampling from a uniform $[-0.2, 0.2]$ distribution, interestingly this tended to only move around in the same space, which helped to demonstrate just how unique the RND policy had learned.

V. Discussion

Our results demonstrate several key insights into the role of intrinsic motivation in RL and robotic grasping. First, we observed that agents trained with dense rewards consistently outperform sparse rewards for both RND and standard agents, which agrees with our expectation and illustrates the problem with standard RL's reliance on dense external rewards. Second, RND agents exhibited novelty-seeking exploratory behaviors, such as moving towards the camera, to maximize prediction error and hence the intrinsic reward while deviating from the immediate objective. While this demonstrates the intrinsic motivation in action, it also points to some limitations and challenges to effectively leverage intrinsic reward for optimal learning. For our relatively simple grasping tasks, RND did not improve performance over standard agents, as intrinsic motivation appears to distract the RND agent from the primary objective when the solution space is small or easily discoverable.

A main challenge of this project is the overall limited time and resources available, as training RL agents requires lots of time and compute resources. These limitations led us to focus on a simplified grasping scenario rather than the more complex and exploratory setting of cluttered environments, which we initially intended to study. Additionally, more hyperparameter tuning or testing other changes such as observation stacking for the feature models could be interesting avenues to investigate, but we couldn't get to them in time.

Future work may extend our setup to more complex environments and tasks where effective exploration is critical, such as cluttered scenes with occluded target objects. These scenarios would better highlight the benefits of intrinsic motivation in guiding exploration and overcoming sparse rewards. Additionally, we aim to investigate the impact of hyperparameter choices and observation modalities on learning performance.

References

[1] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-Driven Exploration by Self-Supervised Prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017. [Online]. Available: https://doi.org/10.1109/cvprw.2017.70

[2] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," *arXiv preprint arXiv:1810.12894*, 2018. [Online]. Available: https://arxiv.org/abs/1810.12894

[3] Y. Zhu *et al.*, "robosuite: A modular simulation framework and benchmark for robot learning," *arXiv preprint arXiv:2009.12293*, 2025. [Online]. Available: https://arxiv.org/abs/2009.12293

[4] A. Mandlekar et al., “Scaling Robot Supervision to Hundreds of Hours with RoboTurk: Robotic Manipulation Dataset through Human Reasoning and Dexterity,” in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Nov. 2019, pp. 1048–1055. Available: https://doi.org/10.1109/iros40897.2019.8968114

[5] A. Boularias, J. Bagnell, and A. Stentz, “Learning to Manipulate Unknown Objects in Clutter by Reinforcement,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 29, no. 1, Feb. 2015. [Online]. Available: https://www.ri.cmu.edu/pub_files/2015/1/AbdeslamAAAI2015.pdf

[6] L. Wu, Y. Chen, Z. Li, and Z. Liu, “Efficient push-grasping for multiple target objects in clutter environments,” *Front. Neurorobot.*, vol. 17, May 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2023.1188468/full

[7] C. Schwarke, V. Klemm, M. van der Boon, M. Bjelonic, and M. Hutter, “Curiosity-Driven Learning of Joint Locomotion and Manipulation Tasks,” in *Proc. 7th Conf. Robot Learn. (CoRL)*, 2023. [Online]. Available: https://openreview.net/forum?id=QG_ERxtDAP-

[8] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling (2012). The Arcade Learning Environment: An Evaluation Platform for General Agents. CoRR, abs/1207.4708. [Online]. Available: https://arxiv.org/abs/1207.4708

VI. Appendix

A. Training Details

These hyperparameters were essentially initialized to be as similar to the original random network distillation paper as possible while also accounting for the continuous dynamics.

Parameter	PPO	PPO+RND
γ_e	0.99	0.999
γ_i	N/A	0.99
Intrinsic Reward Coefficient	0	1
Extrinsic Reward Coefficient	2	2
Log Std	0	0
λ	0.95	0.95
ε	0.2	0.2
Entropy Coefficient	0.001	0.001
Value Coefficient	0.5	0.5
Learning Rate	0.0003	0.0003
# Environments	32	32
Rollout Length	256	256
Max Grad Norm	0.5	0.5

B. Model Comparisons

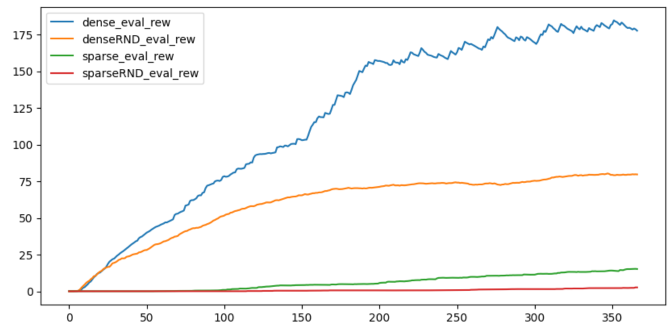

Fig. 8. Overall comparisons between the models (excluding pure RND)

Code: https://github.com/antony-zhao/106b-final-project
Demo Website