Heuristic Learning for Look-Ahead Manipulation Planning in Continuous Action Space

Wissam Bejjani, Mehmet R. Dogar and Matteo Leonetti

I. Introduction

Consider the example shown in Fig. 1 where a robot is tasked with manipulating, using prehensile and non-prehensile planar actions, an object (the orange fruit) to a target location. The robot end-effector has to navigate its way through the clutter whilst not dropping any off the objects of the edges of the surface. Fundamental to the successful execution of such a manipulation task is the ability to reason over the effect of an action on achieving the task goal. Further, it is essential for the actions to be generated from a closed-loop control scheme in real-time for the robot to adapt its motion to the evolution of the interaction sequence with the real world. These two requirements are challenged by the complex interaction dynamics, edge constraints, and variation in object shapes and numbers.

Receding Horizon Planning (RHP) presents itself as an attractive solution that can be run in near real-time when a suitable heuristic is available. RHP works by running multiple stochastic physics-based roll-outs up to a certain horizon in the physics simulator as illustrated by the simulation rendered images in Fig. 1. Each roll-out is evaluated based on the collected rewards during the roll-out and the expected rewards beyond the horizon state: $R_{0:H} = r_1 + \gamma r_2 + \ldots + \gamma^{H-1} r_H + \gamma^H v(s_H)$. RHP returns the *first* action of the best roll-out. In this process, the heuristic is required for guiding the roll-outs towards parts of the state space with higher expected rewards, and for estimating the expected rewards beyond the horizon state.

The goal of this work is to acquire, without prior knowledge that might bias the solution, efficient (in terms of number of actions) manipulation skills that can seamlessly generalize over an arbitrary number of novel objects on a cluttered planar surface in the real world. To achieve this goal, we proposed in [1], [2] interleaving real-world execution with physics-based RHP that is guided by a *learned heuristic*. This approach is based on dynamically mapping the state of the real world to the simulator, where RHP is engaged. The returned action is then executed by the real robot.

To this end, most learning based approaches for manipulation in clutter relies on operating in a discrete action space [3], [2] or well defined action primitives [4]. The achievements of these implementation in challenging high dimensional state spaces was made possible by the relative stability of *off-policy* RL algorithms (ex: deep Q-learning) in discrete actions spaces. However, this often

Authors are with the School of Computing, University of Leeds, United Kingdom {w.bejjani, m.r.dogar, m.leonetti,}@leeds.ac.uk

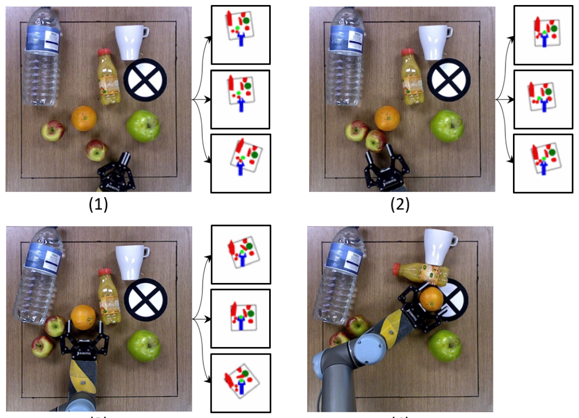

Fig. 1: Real-world execution of moving an object (*orange fruit*) to a target location using image-based Receding Horizon Planning. The abstract images show the predicted horizon states of different stochastic look-ahead roll-outs.

comes at the cost of learning a sub-optimal behavior to what is actually achievable by the robot if it were allowed to use the full *continuous* action space. In our previous work (see https://youtu.be/EmkUQfyvwkY), we observe the robot executing multiple discrete action steps to re-position itself to a specific location while not traversing a significant Euclidean distance in the Cartesian space. Our intuition is that a policy that can operate in a continuous action space would reduce the number of actions required for solving such a manipulation task. In this work present a novel heuristic and its corresponding learning algorithm for operating in continuous action space.

II. State of the Art

There have been multiple recent developments in off-policy value-gradient algorithms for continuous action space (ex: DDPG, CEM-RL, SVG, NAF). Nevertheless, they remain impractical to tune particularly in high dimensional tasks with sparse rewards such as manipulation in clutter. The current state-of-the-art for scalable RL algorithms in continuous state and action spaces is dominated by the TRPO and PPO type of policy gradient methods that are on-policy by nature. They require a moderate level of parameter tuning, and they rely on using large batch sizes and limit on policy updates to reduce gradient variance and by consequence to keep them from diverging.

Even though an on-policy algorithm might be more stable at the cost of high sample efficiency, we still found it hard to tune for the training process to converge in our target application. This motivated us to develop a more stable variant of an actor-critic style RL algorithm with PPO.

III. Heuristic Learning

The performance of RHP is tied to the quality of its heuristic. The heuristic, modeled by a CNN+DNN and parametrized by θ, is trained to estimate the optimal policy with a stochastic policy, as well as the value function of the learned stochastic policy. In applications where the visual input is significant, the motivation for the policy and the value function to share their parameters is to stabilize the learning as they both share the same low level features and learning them together acts as a regularizing element. RHP would use the stochastic policy to sample physics-based roll-outs, and the value function to estimate the expected rewards beyond the horizon states.

Training a randomly seeded Reinforcement Learning (RL) algorithm in cluttered and edge constrained manipulation environment is unlikely to converge as transition samples leading to the goal will not be observed enough many times. Therefore, similarly to our previous work, we divided the training procedure into three consecutive steps:

Data Collection: we collect a large number of demonstration of solving task instances with random parameterization using a kino-dynamic RRT planner.

Imitation Learning: we jump-start the CNN+DNN with the collected sub-optimal demonstrations. The value head of CNN+DNN is trained to approximate the value function of the policy produced by the planner. The policy head of the CNN+DNN is trained to approximate the action distribution from the demonstrations while penalizing a high entropy distribution.

Reinforcement Learning: we implement an actor-critic style algorithm (A2C) with clipped policy updates as defined by the PPO formulation [5]. In a nutshell, the algorithm loops over 1) running the agent in a simulation environment to collect M transition samples. 2) Once the data is collected, the parameters θ of the CNN+DNN are stored as θ_{old}. 3) Then, the value function and the policy are updated together by minimizing the surrogate loss function w. r. t. θ over the M samples in a batch $B = \{\langle s_i, a_i, r_i, s_i' \rangle_i\}$:

$$\begin{aligned}\mathcal{L}_B^{surrogate}(\theta) = \frac{1}{M}\sum_{i=1}^{M} \\ -\ min(r_{ratio}(\theta)A_{adv}, clip(r_{ratio}(\theta), 1-\epsilon, 1+\epsilon)A_{adv}) \\ +\ \beta\ (r(s,a) + \gamma v_{\theta_{old}}(s') - v_\theta(s))^2 \\ +\ \delta\ H_{entropy}(\pi_\theta(.|s)) \qquad (1)\end{aligned}$$

where $r_{ratio}(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$ is the ratio of the probability under the new and old policies, respectively.

$$A_{adv} = r(s,a) + \gamma v_{\theta_{old}}(s') - v_{\theta_{old}}(s)$$

is the estimated advantage. $H_{entropy}$ is added to encourage exploration.

However, it remains that, in this loss function, the baseline $v_{\theta_{old}}$ used to compute the advantage A_{adv} is an approximation of the value of the policy **prior** to $\pi_{\theta_{old}}$. This means that the baseline for updating the policy is always one step behind the policy used to collect the data. This often goes unnoticed as the policy updates are bounded by the *clip* function. In our target application however, even a very small change in the policy can entail a drastic change in the value function, causing what is know as catastrophic forgetting. To overcome this problem, we propose updating the baseline, prior to performing an optimization step over Eq. 1, to better estimate the value of the policy used to collect the data while also refraining from causing a change to the action distribution of this policy. This is achieved by first doing a updated of the value function with a preference for keeping the policy unchanged:

TABLE I: Results in discrete and continuous action space

	Discrete	Continuous
Success rate	95%	97%
Action efficiency	0.22 ± 0.02	0.48 ± 0.12

$$\begin{aligned}\mathcal{L}_B^{baseline}(\theta) = \frac{1}{M}\sum_{i=1}^{M}(r(s,a) + \gamma v_{\theta_{old}}(s') - v_\theta(s))^2 \\ +\ \zeta\ D_{KL}(\ \pi_{\theta_{old}}(.|s)\ ||\ \pi_\theta(.|s)\ ) \qquad (2)\end{aligned}$$

where ζ is a tunable hyper parameter. The first term on the right will update the value function of the policy used to collect the M transition samples. Since the policy head and the value function head share the same network and updating one perturbs the other, the second term on the right penalizes the KL-divergence between the action distribution of the policy used to collect the data and any resulting change in the action distribution that might be induced by the value function update.

IV. Experiments

To assess whether performing the manipulation task in a continuous action space offers any benefits over a discrete action space, we compare the performance of the two in simulation using heuristic guided RHP. We look at two evaluation metrics. 1) The *Success rate* per 100 random task instances (with up to 5 objects), where success is declared when the desired object is manipulated to the target region under 35 action steps and without any of the objects dropping of the surface edges. 2) The *Action Efficiency* looks at how many actions were executed before successfully reaching the goal. It is measured in view of the scene complexity which is represented by the clutter density. It is calculated as $\frac{number\ of\ objects\ in\ the\ scene}{number\ of\ actions\ until\ completion}$. The results, presented in Table I, show that the success rate remains relatively the same between the two. Further, the results confirms our hypothesis that a continuous action space for manipulation in clutter offers a substantial benefit to the action efficiency, which went up from 0.22 to 0.48.

References

[1] W. Bejjani, R. Papallas, M. Leonetti, and M. R. Dogar, "Planning with a receding horizon for manipulation in clutter using a learned value function," in *IEEE-RAS 18th International Conference on Humanoid Robots*. IEEE, 2018.

[2] W. Bejjani, M. R. Dogar, and M. Leonetti, "Learning physics-based manipulation in clutter: Combining image-based generalization and look-ahead planning," *arXiv preprint arXiv:1904.02223*, 2019.

[3] W. Yuan, K. Hang, D. Kragic, M. Y. Wang, and J. A. Stork, "End-to-end nonprehensile rearrangement with deep reinforcement learning and simulation-to-reality transfer," *Robotics and Autonomous Systems*, 2019.

[4] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," *arXiv preprint arXiv:1803.09956*, 2018.

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