Synergies between Pushing and Grasping - Final Report

Maulik Dang

May 15, 2019

Abstract

Grasping an object is a very basic operation performed by humans. In order to effectively emulate human behaviour, a robot must be able to perform efficient grasps. Before a successful grasp operation, a push operation might be needed in order to separate tightly packed or cluttered objects. Previous work focuses on performing just the grasp operation, which fails in scenarios where the objects are tightly packed, since the target object might be out of bounds for the robot gripper. In this project, we aim to use self-supervised deep reinforcement learning to make the Kuka robot learn efficient coordination between the pushing and the grasping action. The learning algorithm rewards the agent when it performs a successful push or a successful grasp operation. This would enable the robot to perform efficient grasp operations by leveraging the outcome of the push operations.

Introduction

Skilled manipulation benefits from synergies between non-prehensile (e.g. pushing) and prehensile (e.g. grasping) actions: pushing can help rearrange tightly packed objects to make room for the robot arm and gripper. Similarly, grasping can help displace objects to make pushing movements more precise and collision-free.

A considerable amount of research has been for planning of both the push and the grasp operation. However, these research studies have not tackled these operations in conjunction. Pushing is traditionally studied for the task of precisely controlling the orientation of an object. This led to scenarios where tightly packed objects, that could have been grasped easily, were not grasped due to their arrangement.

In our approach, we learn joint pushing and grasping policies through self-supervised deep reinforcement learning, unlike prior methods which involved isolated studies and heuristic (and hard coded) supervised learning techniques. Pushing actions are useful only if, in time, enable grasping. The policies are trained using an end-to-end a deep network which is provided with visual observations and outputs expected return (i.e. in the form of Q values) for pushing and grasping actions. The joint policy then chooses the action with the highest Q value – i.e. , the one that maximizes the expected success of current/future grasps.

Related Work

Planning non-prehensile motions, such as pushing, has been a topic of research since the inception of robotic manipulation. Many of the methods rely on modeling assumptions that do not hold in practice. For instance, non-uniform friction distributions across object surfaces and the variability of friction are only some of the factors that can lead to erroneous predictions of friction-modeling pushing solutions in real-world settings. Recent methods have explored data-driven algorithms for learning the dynamics of pushing, but most of these approaches have focused on the execution of stable pushes for a single object. Like Pushing, Grasping has also been well studied in the domain of model-based reasoning.

Combining both non-prehensile and prehensile manipulation policies is interesting and an area of research that has been much less explored. Dogar et al. [1] present a robust planning framework for push-grasping to mitigate the failure rate of grasping operations. They include an additional motion primitive – sweeping – to move around obstacles in clutter. However, the policies in their framework were largely human engineered. Boularias et al. [2] presented an approach that aligns closely to our work. Their work uses reinforcement learning for training control policies to select among push and grasp proposals represented by hand-crafted features. However in order to determine the action with highest expected reward, the approach uses hand engineered features derived from the visual representation of the scene.

Techical Approach

Problem Formulation

The task of pushing-for-grasping is modeled as a Markov decision process: at any given state s_t at time **t**, the agent (i.e. robot) chooses and executes an action at according to a policy $\pi(s_t)$, then transitions to a new state s_{t+1} and receives an immediate corresponding reward $R_{a_t}(s_t, s_{t+1})$ The goal of our robotic reinforcement learning problem is to find an optimal policy π^* that maximizes the expected sum of future rewards, given by

$$R_t = \sum_{i=t}^{\infty} \gamma R_{a_i}(s_i, s_{i+1}),$$

i.e. γ-discounted sum over an infinite-horizon of future returns from time t to ∞. In this work, we investigate the use of off-policy Qlearning to train a greedy deterministic policy $\pi(s_t)$ that chooses actions by maximizing the action-value function (i.e. Q-function) $Q_\pi(s_t, a_t)$, which measures the expected reward of taking action a_t in state s_t at time t. Formally, our learning objective is to iteratively minimize the temporal difference error δ_t of $Q_\pi(s_t, a_t)$ to a fixed target value y_t:

$$\delta_t = |Q(st, at) - y_t|$$

$$y_t = R_{a_t}(s_t, s_{t+1}) + \gamma Q(s_{t+1}, \underset{a}{\operatorname{argmax}}(Q(s_{t+1}, a')))$$

where a' is the set of all available actions.

Method

State Representation

Each state s_t is modeled as an RGB-D heightmap image representation of the scene at time t. RGB-D images from a fixed-mount camera are obtained and the data is projected onto a 3D point cloud, and orthographically back-project upwards in the gravity direction to construct a heightmap image representation with both color (RGB) and height-from-bottom. The edges of the heightmaps are predefined with respect to the boundaries of the robot's workspace for picking.

Actions

Each action is parameterized as a motion primitive behavior ψ (e.g. pushing or grasping) executed at the 3D location q projected from a pixel p of the heightmap image representation of the state s_t:

$$a = (\psi, q) \mid \psi \in \{push, grasp\}$$

The motion primitives for the project are as follows:
Pushing: Denoted a 10cm push in one of k = 16 directions. The trajectory of the push is straight. It is executed in our experiments using Kuka's gripper.
Grasping: Denotes a top-down parallel-jaw grasp in one of k = 16 orientations.

Learning

For the purpose of learning, our Q-function is modeled as two feed-forward fully convolutional networks (FCNs) ϕ_p and ϕ_g; one for each motion primitive behavior (pushing and grasping respectively). Each individual FCN takes in the heightmap image representation of the state s_t as the input and outputs a dense pixel-wise map of Q values with the same image size and resolution as that of s_t, where each individual Q value prediction at a pixel p represents the future expected reward of executing primitive ψ at 3D location q where $q \mapsto p \in s_t$. To simplify learning oriented motion primitives for pushing and grasping, we account for different orientations by rotating the input heightmap s_t into k = 16 orientations (different multiples of 22.5 degree) and then consider only horizontal pushes (to the right) and grasps in the rotated heightmaps. Thus, the input to each FCN is k = 16 rotated heightmaps, and the total output is 32 pixel-wise maps of Q values (16 for pushes in different directions, and 16 for grasps at different orientations). The action that maximizes the Q-function is the primitive and pixel with the highest Q value across all 32 pixel-wise maps:

$$\underset{a'_t}{\text{argmax}}(Q(s_t, a'_t)) = \underset{(\psi,p)}{\text{argmax}}(\phi_p(s_t), \phi_g(s_t))$$

Rewards

Our reward scheme for reinforcement learning is simple. We assign $R_g(s_t, s_{t+1}) = 1$ if a grasp is successful (computed by thresholding on the antipodal distances between gripper fingers after a grasp attempt) and $R_p(s_t, s_{t+1}) = 0.5$ for pushes that make detectable changes to the environment (where changes are detected if the sum of differences between heightmaps exceeds some threshold).

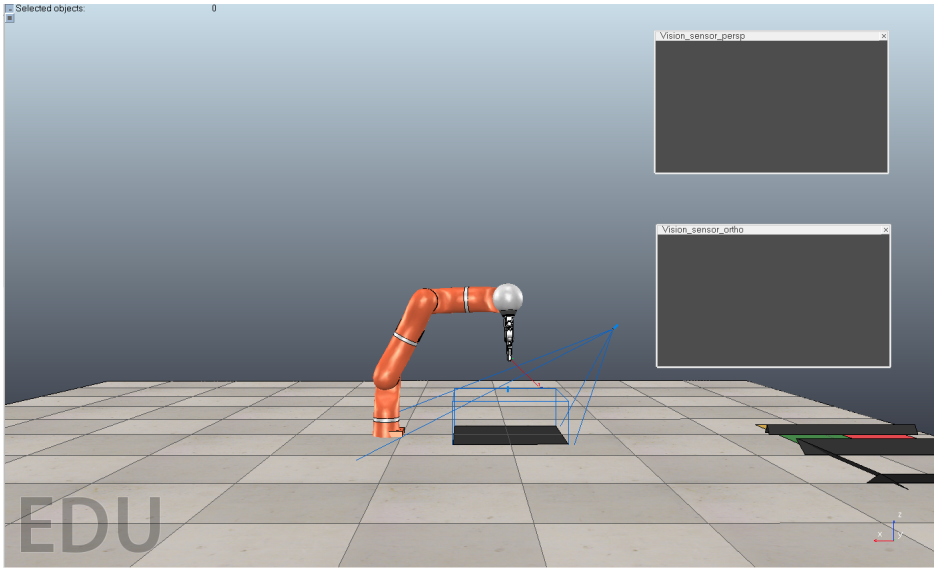

Figure 1: KUKA in a VREP environment

Evaluation

Midterm Goal

The goal decided for our midterm was to get the Kuka robot perform basic pushing, pulling and grasping on a single object.

Status of the Midterm Goal

Since we started with an entirely new Simulation environment (V-REP), we havent been able to achieve the midterm goal fully, but have achieved certain components of the same.

- Able to achieve simple kinematic movements of the Kuka robotic arm. These movements include moving the arm forward, upward and given a reachable point in the robot's workspace, the tip of the arm can reach the aforementioned point. The robot arm tries reaching the point within 60 attempts by modifying its orientation, after which it receives a timeout.
- Able to achieve basic collision detection and avoidance so that the Kuka arm does not collide with itself.
- Able to perform closing and opening of the gripper present on the right arm of the Kuka robot.
- Able to obtain the mesh files and place the objects in the workplace used by the Princeton research team. However, there are some issues with the orientation of the robot and placing the objects.

Status of the Final Goal

- The paper was successfully replicated to work on a KUKA robot arm.
- Due to GPU constraints, the model was not trained to its full capability and was trained on a CPU for 48 hours straight, amounting to approximately 1351 training iterations.
- During each training iteration, 10 objects were placed randomly on the mat, and the KUKA robot tried pushing and grasping operations in order to earn rewards and update its policy.
- Due to the less training, the grasp success rate is around 20% for the robot arm, but it performs arguably well on a collection of small objects and is able to grasp most of the objects.

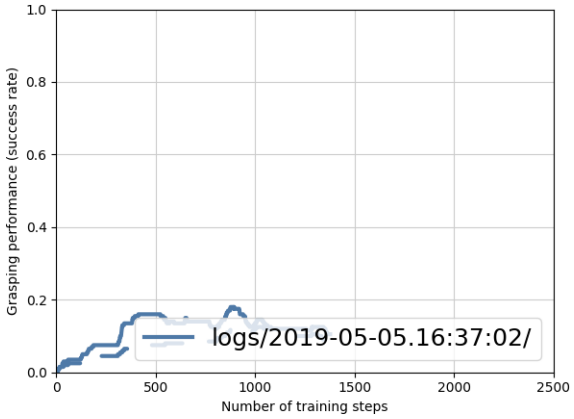

Figure 2: Grasp Success rate wrt Number of training steps

Conclusion

This project aims to enable a KUKA robotic arm to learn the coordination between pushing and grasping operations in order to declutter a set of objects and separate them. The main idea behind

the approach was to use Deep Reinforcement Learning to train the arm using different reward values for ther Push and the Grasp operation. At the end, the project demonstrates that the KUKA robotic arm is capable of learning strategies that would enable synchronization between the pushing and the grasping actions.

However, there are still a few challenges in the approach that are not tackled in this project. At times, the KUKA arm pushes almost all of the objects out of its scope thereby rendering them in a position where a future grasp or a push. Another major roadblock is that training for the project is very GPU intensive and the CPU training time is a lot (around 3 minutes for one action). Lastly, the robot arm tries grasping objects with a wrong orientation of the gripper. For example, trying to grasp really wide objects from the wide side, instead of the other side. Hopefully, these issues could be resolved in the future by implementing hand crafted features or training for more time.

References

[1] M Dogar, Kaijen Hsiao, Matei Ciocarlie, and Siddhartha Srinivasa. Physics-based grasp planning through clutter. 2012.

[2] Abdeslam Boularias, James Andrew Bagnell, and Anthony Stentz. Learning to manipulate unknown objects in clutter by reinforcement. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.