Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning

Zeng, Song, Welker, Lee, Rodriguez, Funkouser

Charles Averill

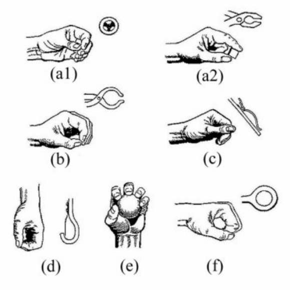
charles@utdallas.edu
Intelligent Robotics and Vision Laboratory
The University of Texas at Dallas

April 2022

Table of Contents

Background

- What does your hand do?
    - Prehensile motion (grasping)
    - Non-prehensile motion (pushing)
- We want robots to perform tasks humans can
- Humans can simplify cluttered environments by pushing and grasping objects
- Therefore, robots should be able to skillfully push and grasp objects to better interact with complex environments



- Pushing - Can rearrange objects to make space for grasping apparatus
- Grasping - Can displace objects to reduce object collisions when pushing

Background

- Pushing and grasping have been studied a lot for the past 40 years, but in isolation
- Previous pushing research typically involves loosely-defined policies
  - "Separate 2 objects"
  - "Make space at this location"
  - "Break up clusters of objects"
- Previous grasping research typically involves human-labeled data - inefficient and prone to overfitting
- Combining pushing and grasping has been researched, but relies on hardcoded policies $\rightarrow$ limited utility, capability to adapt

Motivation

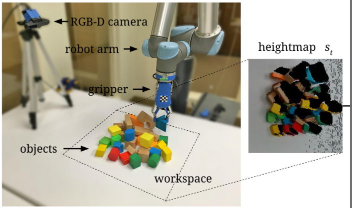
- Combine pushing and grasping learning for enhanced synergy between the two actions
- Utilize reinforcement learning to avoid manually-labeled data
- Produce an end-to-end DNN training framework, no intermediate data processing to improve results

Problem Formulation

- Markov Decision Processes excel in situations with deterministic control in addition to randomness, like robots in the real world
  - At a time t for a given state s_t, the agent's policy $\pi(s_t)$ will choose an action a_t.
  - Executing a_t transitions the agent to the state s_{t+1} and yields the reward $R_{a_t}(s_t, s_{t+1})$
- Goal is to train a policy that maximizes $\sum_{i=t_o}^{t_{end}} R_{a_i}(s_i, s_{i+1})$
- Ideal policy should choose actions such that the action-value function $Q_\pi(s_t, a_t)$ is maximized
  - Action-value function (Q-function) measures expected reward from taking action a_t in state s_t
- This is the general outline of the Q-learning algorithm (if you hadn't guessed already)

Method

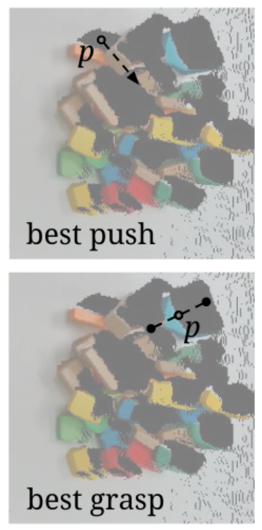
- States s_t are represented by RGBD images of the table scene that are projected into a heightmap
  - Heightmap is rotated in 16 uniform orientations to account for different control angles
- Actions a_t are represented by a tuple of one of the two motion primitives (pushing or grasping) ψ at a 3D location q



$$a = (\psi, q) | \psi \in \{\text{pushing}, \text{grasping}\}, q \cong p \in s_t$$

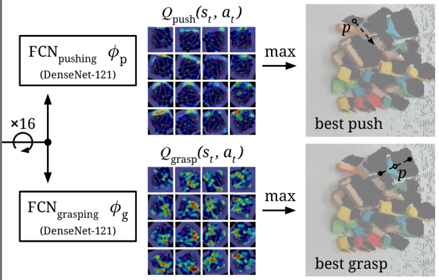
Action Types

- When pushing, q represents the starting point of a straight 10cm push
- When grasping, q represents the center position of a grasp motion in which the gripper attempts to move 3cm below q before grasping



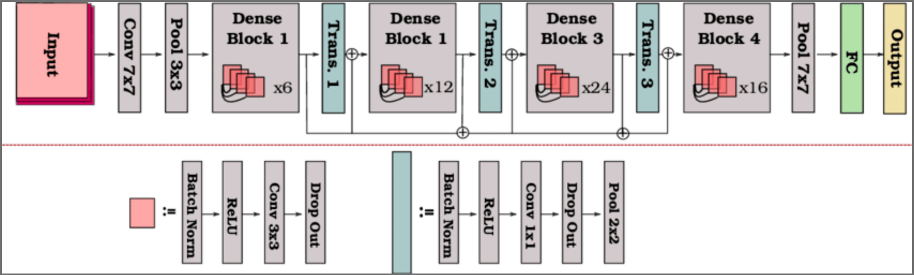
Learning the Action-Value function with CNNs

- Authors utilize two FCNNs, one for each motion primitive
- Networks take in heightmaps (s_t) and output heatmaps of expected Q values for their respective actions at the corresponding q to each pixel p



- Maximum Q value across each network output is used to select the action to perform
- Grasping reward $R_g(s_t, s_{t+1}) = 1$ iff grasp was successful
- Pushing reward $R_p(s_t, s_{t+1}) = 0.5$ iff push caused some change in the heightmaps of s_t, s_{t+1} greater than a threshold of τ.
  - Pushing reward doesn't encode the concept of enabling grasping, just change in the environment.

Network Architecture



DenseNet-121 Architecture Diagram

- Networks have identical structure: two 121-layer DenseNets, pretrained on ImageNet
- One network processes RGB, the other processes depth
- Concatenation and convolution layers appended

Results

- Baseline policies were implemented to measure the novel policy against
  - Reactive Grasping-only Policy - Same problem structure, with a single FCNN
  - Reactive Pushing and Grasping Policy - Same as above, with an additional FCNN, selects action which provides the greatest immediate reward rather than actions that are better long-term
- All three policies were tested in simulated and real environments, with random and challenging arrangements of objects
- The novel policy out-performed the baselines in each scenario by 10-50%

Results

SIMULATION RESULTS ON RANDOM ARRANGEMENTS (MEAN %)

Method	Completion	Grasp Success	Action Efficiency
Grasping-only [8]	90.9	55.8	55.8
P+G Reactive	54.5	59.4	47.7
VPG	**100.0**	**67.7**	**60.9**

SIMULATION RESULTS ON CHALLENGING ARRANGEMENTS (MEAN %)

Method	Completion	Grasp Success	Action Efficiency
Grasping-only [8]	40.6	51.7	51.7
P+G Reactive	48.2	59.0	46.4
VPG	**82.7**	**77.2**	**60.1**

REAL-WORLD RESULTS ON CHALLENGING ARRANGEMENTS (MEAN %)

Method	Completion	Grasp Success	Action Efficiency
Grasping-only [8]	42.9	43.5	43.5
VPG	**71.4**	**83.3**	**69.0**

Conclusion

- Goal was to train FCNNs to predict reward policies that promote pushing and grasping to simplify a cluttered scene
- Networks receive RGBD heightmaps of a scene as input and output a heatmap of reward values after a push or grasp
- Rewards were tuned to promote synergy between pushing and grasping
- Novel policy outperformed baselines by a significant amount

Resources

- Paper: https://arxiv.org/pdf/1803.09956.pdf
- Writeup: https://vpg.cs.princeton.edu/
- Github: https://github.com/andyzeng/visual-pushing-grasping
- Markov Decision Processes: https://en.wikipedia.org/wiki/Markov_decision_process