1 Summary

1. Learns how to combine individually parameterized grasping policy and pushing policy.
2. The 2 policies map directly from visual obs to actions.
3. Both are trained jointly to maximize the number of successful grasps.
4. Propose an interesting discrete parameterization of the action space that enables efficient learning, with translational and rotational invariance and parallelization of action evaluation

2 State Representation

1. Each state is represented as a RGB-D heightmap image representation of the scene.
2. To compute the heightmap, RGB-D images are captured from a fixed camera, project 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 (D) channel.
3. The edges of the heightmaps are predefined with respect to the boundaries of the agent's workspace.
4. Each pixel in the heightmap represents a vertical column of 3D space in the agent's workspace.

3 Actions Representation

1. Each action is a motion primitive behavior (pushing or grasping) executed at the 3D location projected from a pixel of the heightmap image.
2. For pushing, the length of the push is fixed.
3. For grasping, the z value of the grasp configuration (gravity direction) is predefined as a function of the depth corresponding to the pixel.
4. Robot arm motion planning is automatically executed with stable, collision-free IK solvers.

4 Learning Convolutional Action-Value Functions

1. There are 2 Q-function, one for grasping and one for pushing.
2. Each function takes as input the heightmap image and outputs a dense pixel-wise map of Q values with the same image size and resolution as that of the heightmap.
3. Each individual Q prediction represents the value of executing an action at the 3D location corresponding to a pixel.
4. To account for the different orientations grasp and push actions can be executed, they rotate the input heightmap by 16 different orientations and then consider only horizontal pushes and grasps in the rotated heightmap.
5. Each Q-function thus receives as input 16 rotated heightmaps and output 16 pixel-wise maps of Q values, one map for each heightmap.
6. The selected action is the motion primitive at a pixel location that has the highest values across all pixel locations and value functions

$$\text{argmax}_{(\psi,p)} \left(\phi_{\text{p}}\left(s_t\right), \phi_{\text{g}}\left(s_t\right)\right)$$

   where (ψ, p) represents the action. ψ is the motion primitive and p is the location in the heightmap.
7. Their state and action parameterization is thus a pixel-wise parameterization.
8. They argue this allows for using Conv. Net as value function approximators and therefore has several benefits.

9. First, the value prediction for each action now has an explicit notion of spatial locality to neighboring actions and to the state (thanks to receptive fields of the Conv Net).

10. Second, pixel-wise and rotation-wise calculation is fast, requiring a single forward pass through the Conv Net.

11. The parameterization of the action is invariant to translation and rotation, leading to needing less data for the model to converge.

5 Rewards

1. They assign a reward of 1 if a grasp is successful, computed by thresholding the antipodal distance between gripper fingers after a grasp attempt.

2. The reward for a push action is 0.5 for pushes that make detectable changes in the environment, computing by thresholding sum of differences between heightmap.

6 Questions

1. They mention they set the friction coefficient of the simulation so that it is as similar as possible to that of the real world. How did they do this?

2. Their sample efficiency is attributed to their discrete parameterization of the action space. But there is no proof of this it seems.

3. Why did they have to go through the complex transformation to get the heightmap of the scene. Why not just use an over-the-shoulder setup?