Real Robot Challenge 2020 - Phase 3 Report

Team Name: sombertortoise

Abstract: In Phase 3, we use a mid-level policy with three separate low-level primitives to solve the cuboid manipulation task at the four difficulty levels. Each task is decomposed into three stages: moving the fingers to the cube to grasp it, rotating it on the table to minimize orientation error, and reposing it with three fingers. These grasping, lifting, and rotating primitives are sequenced with a state-machine that determines which primitive to execute given the current state, goal, and difficulty level. In our own evaluation, our mid-level policy shows robust performance over multiple runs with randomized initial and goal positions.

1 Introduction

Our approach rests on the idea that in-hand manipulation tasks can be broken down into manipulation primitives [1]. In phase 3, we decompose the task of grasping and reposing the cuboid into three primitives: grasping the cuboid, rotating the cuboid on the table, and finally, re-positioning it. These primitives are planned and executed using model-based trajectory planning and controllers, and sequenced with a state machine. In our tests, we monitored the performance of employing a joint RL-based pusher policy to recenter and reorient the object before employing our model-based controller. However, we found that while it did at times speed up reorientation on the table, the model-based method alone was more robust (and could more consistently get a good score) to use alone in the final submission. Example runs for each of the four levels can be seen in this video.

2 Method

2.1 Overview

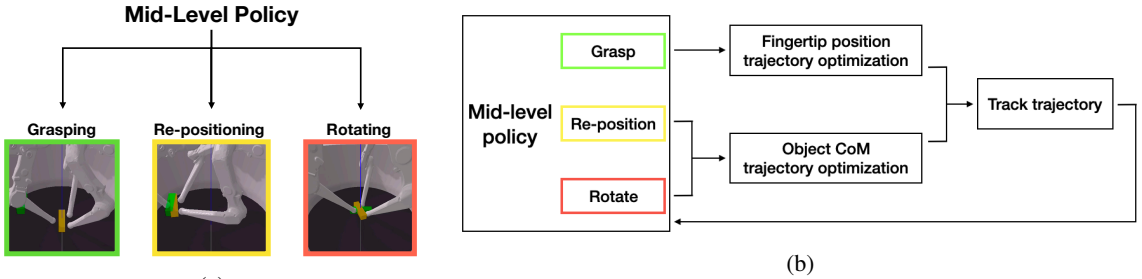

Figure 1: (a) Manipulation primitives used by the mid-level policy (b) State machine that chooses, plans, and executes primitives.

Fig 1 shows our overall approach. Using a hand-designed state machine, the mid-level policy sequences grasping, rotating, and re-positioning primitives in order to move the cube to the goal position. For all the tasks, we assume prior knowledge of robot dynamics, physical parameters of the object, as well as accurate state feedback on object pose and finger joints. With that, a model-based trajectory optimizer and controller are sufficient for implementing all of the primitives. For each primitive, we compute the desired trajectories for each fingertip in Cartesian space using trajectory optimization and track these trajectories using an impedance controller. For re-positioning and rotating the object, we first compute a trajectory for

the object's pose, and then use this trajectory to compute the desired trajectories that the fingertips must follow. To complete each task, the mid-level policy first grasps the cube, and then determines whether to re-position or first rotate the cuboid.

We additionally trained an RL policy in parallel to push and rotate the cuboid on the table, since such trajectories are contact-rich and are difficult to plan for. While this policy was able to perform the rotation task in simulation, due to challenges faced in bridging the gap between simulation and the real robot, we in the end opted to use the impedance controller alone as it was more robust and had less variance in the scores for the level 4 task.

2.2 Controller

We use a PD controller to track the desired fingertip position trajectories. We compute the joint torques necessary for tracking the desired fingertip position and velocity trajectories in Cartesian space using an PD law similar to the one used in [2] with additional gravity compensation, shown in equation 1. For each finger j, $\Gamma_j \in \mathbb{R}^3$ is the output torques of one finger, f_j is the desired force to be applied by the finger, $J_{v,j}$ is the linear Jacobian of one fingertip, and G_j is the gravity compensation vector.

$$\Gamma_j = J_{v,j}^T(k_p(x_{d,j} - x_j) + k_v(\dot{x}_{d,j} - \dot{x}_j) + f_j) + G_j \tag{1}$$

2.3 Grasping

To grasp the cuboid securely, we use a three-fingered grasp to pinch the cuboid across its short axis, as shown in Figure 1a. This grasp is used for both the re-position and rotation primitives. Contact points are chosen to be reachable by each finger, based on the initial pose of the object. We formulate a direct collocation trajectory optimization problem to compute collision-free fingertip position and velocity trajectories to desired contact points on the cube. The optimization problem considers only the kinematics of the fingers and assumes that the pose of the cube is static. To prevent the fingers from colliding with the cuboid, we include a collision penalty term in the objective function.

2.4 Rotating and re-positioning

Assuming fixed contact point positions, we use direct collocation to compute an open-loop trajectory for the object as well as the associated forces that need to be applied at the contact points to move the object along this trajectory. In the optimization problem, we constrain the contact forces to lie within linear approximations of friction cones to prevent contact slippage and specify a target normal force that the fingers should apply to ensure sufficient internal forces on the cube.

In comparison to re-positioning the object, re-orienting the object is considerably less straight-forward, because while re-positioning the cuboid can be achieved with a single grasp, rotating the cuboid could potentially require using multiple grasps. To account for this, we introduce additional logic into our state machine to use multiple grasps. Given the difficult nature of re-orienting the object in mid-air, we only attempt to rotate the object while it rests on the table. To insure that the fingers remain inside their configuration space while rotating the cuboid, we rotate the object in forty-five degree increments, resetting and re-grasping the object between each rotation.

3 Results and discussion

Our current method is able to rotate the cuboid on the table and re-position it to goal position. We evaluated our method on each level with several goals, running three jobs for each goal. Table 1 shows the mean and

	Level 1	**Level 2**	**Level 3**			**Level 4**		
Goal	1	1	1	2	3	1	2	3
Mean score	-3434	-8324	-28298	-8372	-22389	-48588	-24392	-36300
Median score	-3455	-7889	-12241	-8676	-11775	-49773	-26250	-39230

Table 1: Mean and median scores across different goals/levels

median scores for each goal. As expected, the scores primarily depend on the goal pose; goal poses closer to the center of the arena where the cuboid starts result in better scores. However, during testing, we also observed that the varied initial poses of the cuboid also caused large variations in scores across runs for the same goal pose. This was particularly true for level 4 goals, since the initial orientations varied greatly.

Although our method is able to complete levels 1, 2, and 3, as well as make good progress in level 4, there are several improvements that could be made, in the future. Firstly, our method relies on accurate knowledge of object pose, and is therefore not robust to the jittery object orientation measurements. Secondly, our method still suffered from steady state errors when re-positioning the cuboid, which could be improved by taking into account object pose feedback with either another feedback law or a learned policy. This includes some of the hybrid control approaches we experimented with on this hardware platform, such as combining RL with impedance control in a hierarchical or residual policy. Finally, more sophisticated reasoning could be used to choose grasps and sequence rotations for re-orienting the object.

References

[1] A. M. Okamura, N. Smaby, and M. R. Cutkosky. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 255–262. IEEE, 2000.

[2] M. Wüthrich, F. Widmaier, F. Grimminger, J. Akpo, S. Joshi, V. Agrawal, B. Hammoud, M. Khadiv, M. Bogdanovic, V. Berenz, et al. Trifinger: An open-source robot for learning dexterity. *arXiv preprint arXiv:2008.03596*, 2020.