Real Robot Challenge Phase 3: Manipulating Objects using High-Level Coordination of Motion Primitives

Team: **troubledhare**

December 13, 2020

Abstract

In Phase 1, we presented an approach to rigid-body manipulation based on carefully-designed motion primitives. In Phase 2, we demonstrated that these primitives combined with a more robust state machine can reasonably complete all 4 tasks on a real robot. For Phase 3, to perform manipulation with a cuboid, we continue with the same approach with a different set of motion primitives. Our approach shows promising results on the position-based tasks for cuboid manipulation. In general, our approach emphasizes the use of ML machinery only when classical approaches fail.

Phase 3: Manipulating a Cuboid

Jacobian Force Controllers and PID Loops As in Phase 1 and 2, we first reduced the more complicated joint space $\mathbf{q} \in \mathbb{R}^9$ (3 fingers $\times$ 3 joints per finger) into the more interpretable space $\mathbf{x} \in \mathbb{R}^9$ 3D Cartesian position of each end effector, discarding finger orientation due to the rotational near-symmetry of the end effectors. We retrieve the Jacobian matrix calculated from PyBullet: $\mathbf{J} := \frac{\partial \mathbf{x}}{\partial \mathbf{q}}(\mathbf{q}, \dot{\mathbf{q}})$. We can use its inverse to convert a task space force command into a joint torque command: $\dot{\mathbf{q}} = \mathbf{J}^{-1}\dot{\mathbf{x}}$. $\mathbf{J}$ may be singular or non-square, so we use its damped pseudo inverse to guarantee a good solution at the cost of slight bias: $\dot{\mathbf{q}} = \mathbf{J}^\top \left(\mathbf{J}\mathbf{J}^\top + \lambda \mathbf{I}\right)^{-1} \dot{\mathbf{x}}$. We combine this with gravity compensation torques from PyBullet's inverse dynamics module to command gravity-independent linear forces at each finger tip.

We build upon these linear force commands to create position-based motion primitives. Given a target position for each finger tip, we can construct a feedback controller with manually-tuned PID gains (see Figure 1). Since these systems are linear if we avoid collisions, multiple problems can be solved simultaneously through superposition.

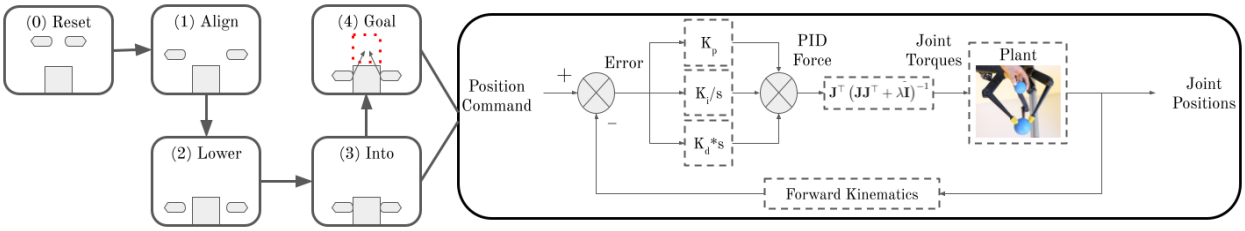

Figure 1: State Machine and Inner PID Loop. On interim failure (such as a time-out or dropped cuboid), all states will return to state 0 (*Reset*).

State Machine Finally, we combine a series of carefully-designed motion primitives into a simple finite state machine.

0. *Reset:* Move arms to a pre-defined position well above the cuboid. Based on the object orientation, set a resting finger which always remains in the reset position.

1. *Align:* Designate 2 contact points on the opposite longer side faces of the cuboid. Move each finger to a point above these grip points (to avoid cuboid collisions).

2. *Lower:* Lower the fingers to the grip points.

3. *Into:* Initiate force closure by commanding a constant force towards the center of the cuboid at each finger until both finger tip force sensors are tripped.

4. *Goal:* Set the target pose to be the current pose plus the difference between the current cuboid location and the target cuboid location. Superimpose this onto the force from *(3)*.

In our final submission, for all difficulty levels, *Goal* continues indefinitely. If the robot spends too long in any given state, or the cuboid drops out of its fingers, it will return to *Reset* and retry.

Results and Discussion In comparison to our performance in Phase 2, we observe a slight degradation in performance for this phase. While we were not able to perform thorough experiments for each of the difficulty levels, we observe some consistent failure cases that lead to the reduction in the average reward accumulated. First, due to noisy pose observations and uncertainty in joint positions, we notice grasping failure. This did not affect our approach that much when manipulating the cube, due to the higher margin for error. Second, in case of a successful grasp, there are cases where the cuboid slips through both the fingers. This issue can be addressed by more diligent PID tuning or alternatively considering systems like MPC or iLQR [1].

For Level 4, we wanted to deploy an additional premanipulation state machine which minimizes the error in Yaw., similar to how we did in Phase 2. Upon reaching the goal position, for the orientation errors in Pitch and Yaw, we planned on using the third finger to account for large errors in Pitch and orthogonal pushing by the other two fingers for small errors in Yaw. Due to time constraints, we were unable to implement these additional motion primitives.

Conclusion

We implemented a method of manipulation that utilizes a set of carefully engineered position-based motion primitives and Jacobian-based feedback control. Our approach worked fairly well in the first 2 phases of the challenge and has a graceful degradation of performance in Phase 3. In future, we intend to run some rigorous experiments with more complex objects on the TriFinger simulator. We plan on having an online learning algorithm that can select the optimal motion primitive given information about the object. Previous work done for the manipulation of deformable objects has shown promise using this contextual bandit / restricted RL approach [2].

1 References

[1] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," *ICINCO (1)*, 2004.

[2] E. K. Gordon, X. Meng, M. Barnes, T. Bhattacharjee, and S. S. Srinivasa, "Adaptive robot-assisted feeding: An online learning framework for acquiring previously-unseen food items," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.