Real Robot Challenge 2021: Cartesian Position Control with Triangle Grasp and Trajectory Interpolation

Rishabh Madan  
Cornell University  
rm773@cornell.edu

Harshit Sikchi  
The University of Texas at Austin  
hsikchi@utexas.edu

Ethan K. Gordon  
University of Washington  
egkordin@cs.washington.edu

Tapomayukh Bhattacharjee  
Cornell University  
tapomayukh@cornell.edu

Abstract: We present our runner-up approach for the Real Robot Challenge 2021. We build upon our previous approach used in Real Robot Challenge 2020 [1]. To solve the sequential goal-reaching task introduced in RRC 2021, we focus on two aspects to achieving near-optimal trajectory: Grasp stability and Controller performance. In the RRC 2021 simulated challenge, our method utilizes a hand-designed Three-Jaw Chuck Grasp combined with Trajectory Interpolation for better stability during the motion for fast goal reaching. In Stage 1, we observe reverting to Triangle Grasp, which was used in the 2020 Edition, provides a more stable grasp when combined with Trajectory Interpolation. The video demonstration for our approach is available at https://youtu.be/dlOueoaRWrM. The code is publicly available at https://github.com/madan96/benchmark-rrc.

Keywords: Manipulation, Grasping, Real Robot Challenge

1 Introduction

Dexterous manipulation, although an everyday task for humans, is a challenging task in the robotics community. Some of the prior successes [2, 3, 4, 5] come with prohibitive engineering costs. Real Robot Challenge (RRC) presents a testbed for roboticists worldwide to remotely experiment with their method on robust real-world robotic systems and thus promotes democratization of robotics, particularly robot manipulation. The environment in Stage 1 of RRC 2021\(^1\) consists of a three-finger manipulator tasked with grasping a cube and moving it in a sequence of waypoints. The final stage uses the three-finger manipulator to arrange several small dices in a specific pattern.

Figure 1: Environments from the RRC 2021. Left: Manipulating a cube along a sequence of waypoint poses, Right: Arranging the dices in a specific pattern.

In this report, we present a practical method for solving the first task of the challenge: manipulating the cube along a waypoint trajectory. We identify an issue in our previous approach [1] which

\(^1\)https://real-robot-challenge.com/protocol
comprised of a Cartesian Position Controller with the Triangle Grasp: when the waypoint changes the controller accumulates error rapidly and the grasp used for manipulating the cube for the current waypoint might not be well suited for the next waypoint. To resolve this issue, we propose a fix using trajectory interpolation (see Figure 2) – we rediscretize the waypoints in a finer scale that leads to smoother motions with added grasp stability. We empirically test different grasping mechanisms and present an approach that achieved the runner-up position in Real Robot Challenge (RRC) 2021.

2 Background

2.1 Problem setting

We consider the first stage of the RRC 2021 challenge, which consists of a TriFinger manipulator tasked with moving the cube along a sequence of waypoints specified in the trajectory. The observations in the environment are composed of robot joint positions, velocities and torques along with the cube 6D pose and the desired goal position for the cube. The environment action interface allows us to control the robot via position control, torque control or both. The reward in the environment is the negative of distance to the current waypoint scaled by the range of motion available in each dimension. Precisely,

$$r(o, g) = -\frac{\|o_{\text{xy,cube}} - g_{\text{xy}}\|_2}{\text{range}_{\text{xy}}} + \frac{\|o_{\text{z,cube}} - g_{\text{z}}\|}{\text{range}_{\text{z}}}$$

(1)

where \(o\) is the observation, \(g\) is the goal location of the cube, \(\|o_{\text{xy,cube}} - g_{\text{xy}}\|_2\) is the euclidean distance in the \(x - y\) plane of the cube’s current location and the goal location.

2.2 Cartesian Position Control (CPC)

CPC involves reducing the control problem in the joint space of all three fingers to performing control in the 3D \((x, y, z)\) Cartesian space of the end effector for each finger. We use Pybullet to obtain the Jacobian matrix \(J := \frac{\partial f}{\partial q}(q, \dot{q})\) and the damped pseudo inverse of \(J\) to ensure a good solution to joint velocity \(\dot{q} = J^\top (JJ^\top + \lambda I)^{-1} \ddot{x}\). This is combined with the gravity compensation torques to command gravity-independent linear forces at each joint. These linear force commands are then used with a PID controller to construct position-based motion primitives. The controller takes a target Cartesian goal position and minimizes the primitive-specific error between the current and target position.

2.3 Grasping Strategy

We experiment with two different grasps: Three-Jaw Chuck Grasp and Triangle Grasp. Three-Jaw Chuck Grasp involves holding the cube using two opposite vertical faces by placing the end effector of one of the fingers on one side (as a thumb) and the other two end effectors on the opposite side (as the index and middle finger). Triangle Grasp entails placing the three end effectors on three of the four vertical faces of the cube such that they connect to form an equilateral triangle.
2.4 Trajectory Interpolation

The PID errors show steep deviations in values when directly switching from the current waypoint to the next waypoint. Inspired by traditional industrial robots, we implement linear trajectory interpolation in the Cartesian space, which samples \( N \) equidistant points along the straight line connecting the current and next waypoint. This allows for smoother transitions between waypoints, thus improving the grasp stability due to reduced in-hand movement of the cube.

3 Method

3.1 Pre-Stage: Solving the challenge in the simulated environment

RRC 2021 challenge consists of a pre-stage challenge which is a simulated version of the Stage-1 challenge. This simulated environment allows for rapid prototyping and testing and is naturally used as a testing ground for our method before we move on the real-world challenge. To solve the prestage challenge we rely on CPC for manipulation from the previous version of the challenge RRC 2020 [1] and test various different grasping strategies like Triangle Grasp, Three-Jaw Chuck Grasp and Parallel grasp [6] in this environment. This environment differs from the previous work [1] as previous work deals with manipulating the cube to a single goal locations whereas our task is to manipulate the cube to a number of different goal locations. This necessitates grasping strategies that are robust to manipulator motions when reaching the immediate goal and consequent goal changes. Our experiments are built upon the open-source repository for the RRC 2020 challenge.

Our experiments show that with the naive strategy of grasping the cube and manipulating with CPC, the robot frequently drops the cube when the active goal changes. This poor behaviour arises due to two major factors: (a) sudden jerk that the fingers experience due to a PID controller error accumulation and often-times attempts of fast-switching to active goal, (b) a grasp that is poorly conditioned for different goals. These effects result in poor grasping and ultimately the cube slipping out.

We experiment with two strategies to solve this issue. First, we switched to Three-Jaw Chuck Grasp that allows for a more stable grasp across different goal locations when shifting from one goal position to another. Second, we implement a linear trajectory interpolation mechanism to generate more fine-grained intermediate waypoints or subgoals in the direction vector of the active goal, thus allowing for a smoother transition to the active goal using a PID controller. Although the Three-Jaw Chuck Grasp had high performance, we identified a potential shortcoming of Three-Jaw Chuck Grasp that it fails to regrasp dropped cube along the arena perimeter. This shortcoming of Three-Jaw Chuck Grasp can be rectified by using a Triangle Grasp, which empirically provides a more robust mechanism to lift the cubes near the outer perimeter of the arena.

Stage 1: Solving the challenge with Real Robot

In this part of the challenge, we began experimentation on the real robot using our approach from pre-stage. After a few rounds of experiments and tuning cycles, we observed that, just as in simulation, the quality of the Three-Jaw Chuck Grasp degrades severely along the arena perimeter. Although we experimented with switching to a Triangle Grasp near the perimeter as proposed in pre-stage, Three-Jaw Chuck Grasp still negatively affected our performance by failing to grasp dropped cube. This difference in performance across simulation and real robot might be attributed to a sim2real gap present in the domain. This motivated us to switch to the Triangle Grasp approach throughout the arena, which led to more stable grasps with rare drops. We then replaced Three-Jaw Chuck Grasp with Triangle Grasp and our the experiments. CPC-TG with Trajectory-Interpolation proved to be a surprisingly effective approach for this sequential goal reaching task.

4 Results

We evaluate a total of 8 real robot experiments with differing goal trajectories, spanning across three different robots.

---

2https://github.com/cbschaff/benchmark-rrc
3Example of failed Three-Jaw Chuck Grasp near the perimeter https://youtu.be/RKZltgcjauY
We observe Triangle Grasp to perform more robustly with fewer cube drops and obtaining higher cumulative reward. This is attributed to the sim2real gap where we observed that Triangle Grasp provided a more stable grasp qualitatively compared to the Three-Jaw Chuck Grasp (CPC-CG) that worked best in simulation.

| Test Environment | mean     | median   | stddev   |
|------------------|----------|----------|----------|
| Simulation (CPC-CG) | -7270.249 | -6113.044 | 2799.856 |
| Real (CPC-CG) | -17071.5  | -17299.9 | -4106.18 |
| Real (CPC-TG) | -9981.3  | -9718.5  | -1142.9  |

Figure 3: An example run of our method on an auto-generated sequence of goals. The objective is to match the goal position under any orientation. Video link: https://youtu.be/dlOueoaRWtM

Figure 4: Grasping along the perimeter of the arena is difficult using the CPC along with the triangulated grasp. Video link:https://youtu.be/RKZltgcjauY

Figure 3 shows our method’s performance qualitatively on an auto-generated sequence of tasks. The method switches between goal smoothly and completes the task with a high cumulative reward compared to other methods submitted as part of the competition\(^4\). Figure 4 shows the failure case of our method where the robot is unable to grasp the object when it is close to the perimeter of the arena. We hypothesize that extrinsic dexterity can play a useful role in regrasping the cube in such situations and achieve even higher return. We leave this avenue to future work.

**Discussion** For the given task-environment, to achieve high rewards, the optimal strategy is to reach the current goal in the shortest path as fast as possible. Such optimal strategy requires having a stable grasp that minimizes cube drops and a controller that allows for quick transitions to goal. Overall our approach shows that manipulating a cube to reach various configurations can be effectively achieved through a simple PID controller with few key considerations of grasp and trajectory. Our method also transfers effectively to real robots requiring less than two hours of finetuning. It is interesting to draw comparisons on this particular task to learning-based methods like RL used in this environment [7],\(^4\)

\(^4\)https://real-robot-challenge.com/leaderboard. Our method was submitted under the alias decimalswift
which requires designing a shaped reward function and then dealing with the sim2real gap, both of which still require significant resources and human effort.
References

[1] N. Funk, C. Schaff, R. Madan, T. Yoneda, J. U. D. Jesus, J. Watson, E. K. Gordon, F. Widmaier, S. Bauer, S. S. Srinivasa, T. Bhattacharjee, M. R. Walter, and J. Peters. Benchmarking structured policies and policy optimization for real-world dexterous object manipulation, 2021.

[2] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In Conference on Robot Learning, pages 1101–1112. PMLR, 2020.

[3] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. The International Journal of Robotics Research, 39(1):3–20, 2020.

[4] In-hand turning. https://www.youtube.com/watch?v=xu5VvyjDLRY,[Online; Retrieved on 24th September, 2020].

[5] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In 2017 IEEE international conference on robotics and automation (ICRA), pages 3389–3396. IEEE, 2017.

[6] M. R. Cutkosky and R. D. Howe. Human grasp choice and robotic grasp analysis. In Dextrous robot hands, pages 5–31. Springer, 1990.

[7] R. McCarthy, F. R. Sanchez, Q. Wang, D. C. Bulens, K. McGuinness, N. O’Connor, and S. J. Redmond. Solving the real robot challenge using deep reinforcement learning. arXiv preprint arXiv:2109.15233, 2021.