Real Robot Challenge Stage 1: Cartesian Position Control with Triangular Grasp and Trajectory Interpolation

Team: decimalswift

July 30, 2021

Abstract

We present an extension to our runner-up approach from the previous edition of the Real Robot Challenge. To solve the task of sequential goal reaching we focus on two aspects to achieving near-optimal trajectory: Grasp stability and Controller performance. In the simulated challenge, our method relied on a hand-designed Pinch grasp combined with Trajectory Interpolation for better stability during the motion for fast goal reaching. In Stage 1, we observe reverting to Triangular grasp to provide a more stable grasp when combined with Trajectory interpolation, possibly due to the sim2real gap. In future stages, we propose using sampling-based planning methods and multi-agent pathfinding algorithms for complex rearrangement tasks. The video demonstration for our approach is available at https://youtu.be/dlOueoaRWrM.

Background: Pre-Stage

In Pre-Stage, we utilized the Cartesian Position Control with triangulated grasp (CPC-TG) from [1] and adapted the publicly available codebase[1] to work with the new task-environment. However, upon directly running CPC-TG in simulation, we observed that it frequently drops the cube when the active goal changes. This poor behaviour arises due to the sudden jerk that the fingers experience and often-times attempts of fast-switching to active goal, both of which result in poor grasping and ultimately the cube slipping out. As part of our solution, we switched to pinch grasp [2] for a more stable grasp when shifting from one goal position to another. In addition, we also implemented a simple linear trajectory interpolation mechanism to generate more fine-grained intermediate waypoints in the direction of the active goal, thus allowing for a smoother transition to the active goal using a PID controller. Although the pinch grasp had high performance, we identified a potential shortcoming of Pinch Grasp that it fails to regrasp dropped cube along the arena perimeter and proposed a solution to utilize triangular grasps when lifting a cube near the corner.

Stage 1: 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 pinch grasp degrades severely along the arena perimeter. Although we experimented with switching to a Triangular grasp near the perimeter as proposed in pre-stage, pinch grasp still negatively affected [2] our performance by failing to grasp dropped cube. This motivated us to switch to the triangular grasp approach throughout the arena, which led to more stable grasps with rare drops. We then replaced pinch grasp with triangular grasp and reran the experiments. CPC-TG with trajectory interpolation proved to be a surprisingly effective approach for this sequential goal reaching task.

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

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

We observe Triangular Grasp to perform more robustly with fewer cube drops and obtaining higher rewards. This is attributed to the sim2real gap where we observed that Triangular Grasp provided a more stable grasp qualitatively.

The video demonstrating our approach on real robots can be found here: https://youtu.be/dlOueoaRWrM. Another video https://youtu.be/RKZltgcjauY shows poor regrasping demonstrated by our baseline (CPC-PG) near arena perimeter.

[1]https://github.com/cbschaff/benchmark-rrc

[2]Example of failed pinch grasp near the perimeter https://youtu.be/RKZltgcjauY

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, which requires designing a shaped reward function and then dealing with the sim2real gap, both of which still require significant resources and human effort.

Stage 2: Rearrange Dice

While we do not yet have the details on this phase, we can continue our philosophy of starting with the simplest approach we can think of that could potentially solve the problem, adding complexity only to increase performance and address shortcomings. One simple approach here is to begin by using all three arms to push all the dice to the rim of the container. This will ensure that no dice exist in collision with one of the target locations. From here, the problem reduces an instance of *Stage 1 Cube Manipulation* $*N$ (the number of dice) with obstacle avoidance. Where, for each die, we use 2 or 3 fingers to execute a grasp, life vertically, move over the target location, rotate to the correct polar orientation (possible in-hand due to at-most a $45°$ rotation assuming no side preference), and dropping the cube into place. This will by no means produce a speedy performance, but it will create a good baseline.

We can begin to augment this approach by drawing from the Multi-Agent Pathfinding Literature (overview at [3], algorithm survey at [4]). At its simplest, we can sample discrete points (including the die start points and all target goal points) across the environment, connecting them into a planar graph. Using 2-3 fingers to push one die at a time and assigning each die a goal to minimize total Euclidean distance traveled, we could solve the planning problem with any graph search algorithm, such as Space-Time A$*$ [5]. Using one finger each, we could potentially move up to 3 dice at a time. Existing off-the-shelf Python implementations [6] can lead to quick iteration cycles here.

1 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] M. R. Cutkosky and R. D. Howe, “Human grasp choice and robotic grasp analysis,” in *Dextrous robot hands.* Springer, 1990, pp. 5–31.

[3] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, E. Boyarski, and R. Bartak, “Multi-Agent pathfinding: Definitions, variants, and benchmarks,” June 2019.

[4] E. Lejeune and S. Sarkar, “Survey of the Multi-Agent pathfinding solutions,” Jan. 2021.

[5] D. Silver, “Cooperative pathfinding,” *Aiide*, 2005.

[6] E. Lejeune and S. Sarkar, “Survey of the multi-agent pathfinding solutions,” 2021. [Online]. Available: http://rgdoi.net/10.13140/RG.2.2.14030.28486