Real Robot Challenge Phase 2

ardentstork

November 13, 2020

Abstract

Our approach is based on residual policy learning. We create a hand-designed control policy and learn residual corrections to this policy. The hand-designed policy consists of a position controller that follows a planned motion and a torque controller that acts to apply corrective forces to the cube. We try a series of grasps until we find one that is able to plan to the goal location. For Task 4, we additionally perform a set of scripted manipulations to align the orientation of the cube with the goal before executing our policy. Due to time constraints, our approach is only slightly changed from Phase 1 of the competition and we use the residual models trained for phase one without modification on Levels 2, 3, and 4. See the future work section for details about improving our training pipeline for the real system.

1 Method

Our approach is based on the success of residual policy learning [1, 5]. Often in robotics it is easy to create a hand-designed policy that solves the task at-hand with a reasonable success rate. Residual policy learning then learns corrections to this policy that improve performance based on some metric or reward function. This often simplifies the learning problem significantly by removing the need for significant exploration outside of the neighborhood around the initial (hand-designed) controller, and leads to a policy that outperforms both the hand-designed policy and standard reinforcement learning techniques while being significantly more sample efficient.

As in Phase 1, we experiment with two different controllers: A force control policy similar to Wüthrich et al. [8] and an extension that includes motion planning. We will first describe these policies and then describe our implementation of residual policy learning.

1.1 Grasping

Both of our hand-designed policies require the cube to be grasped. To do this, we select grasps that meet two properties: the grasp is *feasible*, meaning the finger tips can reach the grasp position and there are no collisions between the fingers (excluding the tips) and the cube, and the grasp is in *force closure*, meaning we can resist any external force applied to the cube. We consider both random grasps and a set of heuristic grasps consisting of ”center of 3” grasps with each finger placed at the center of a face, and ”pinch” grasps with one finger at the center of a face and the other two fingers placed on the opposite face.

In each episode we sample a feasible grasp from this set. To avoid unintentionally hitting the cube due to noise in its pose estimate, we plan to a conservative pre-grasp pose above the cube and then grasp the cube with a scripted motion.

1.2 Force Control

Our force control pipeline is similar to the pipeline described by Wüthrich et al. [8]. We use a PD controller to turn cube pose errors into a wrench applied to the cube's center of mass. Given a grasp of the cube and its resulting contact points, we then solve for tip forces to apply in order to achieve that wrench while maintaining friction cone constraints (see Murray et al. [2] for details). With tip forces, we combine Jacobian transpose control and gravity compensation via inverse dynamics to get joint torques.

Algorithm	Level 1	Level 2	Level 3	Level 4
Force Control (FC)	-138 ± 143	-532 ± 402	-458 ± 477	-1421 ± 345
FC+Residual Learning (RL)	$\mathbf{-74 \pm 29}$	$\mathbf{-123 \pm 69}$	$\mathbf{-122 \pm 54}$	-1527 ± 425
FC+Motion Planning (MP)	-166 ± 178	-177 ± 51	-226 ± 257	-1127 ± 509
FC+MP+RL	-156 ± 82	-171 ± 51	-170 ± 83	$\mathbf{-999 \pm 387}$

Table 1: Mean and std deviation of rewards in simulation at the end of Phase 1 for each algorithm considered. The results are based on 30 replayed episodes using the Phase 1 submission system.

1.3 Incorporating Motion Planning

We extend the above approach by planning a motion of the cube to the goal pose. Once a grasp has been selected, we run an RRT in task space (i.e., the pose of the cube) and assume the grasp is maintained throughout the motion. Using the fixed grasp and a given cube pose, we can solve for the pose of the robot and check for feasibility and collisions. If no feasible path to the goal is found, we select another grasp and try again. To ensure we eventually find a feasible path, we slowly increase the size of the goal set of the RRT. We then use the resulting way points for position control. This is added on top of the force control using the provided torque and position action space.

One downside of this approach is that we assume the grasp is maintained throughout the planned motion. This was mostly true in phase 1, but in phase 2 we found that noise in the cube's pose estimate often causes some slippage. We combat this in two ways. First we reduce slippage by adjusting target fingertip positions to be slightly inside the cube, creating an additional grasping force. And second, we fix slippage errors by adaptively appending corrections to our planned path to drive the position error of the cube to zero.

1.4 Initial Steps for Task 4

Due to a large difference in orientation, many goal poses in level 4 cannot be reached with a single grasp, and therefore the above pipeline will struggle. To solve this, we try to find an initial cube pose which has a small orientation error with respect to the goal pose. This is done by first moving the cube to the center of the work space and performing a sequence of scripted flips and rotations with predefined grasps. Once this is complete, we find that the above pipeline is able to perform reasonably well.

1.5 Residual Policy Learning

We use Proximal Policy Optimization [4] (PPO) to learn corrections to the above policies. We fix the policy to zero for the first 1M timesteps and only train the value function. This allows us to get a good estimate of the value of the hand-designed policy before we start learning. Additionally, we found that the initialization of the policy network was extremely important. Initializing the network to output action distributions with zero mean and small standard deviation greatly improved training as the combined policy stays close to the base policy early in training. Our reward function consists of 3 terms: The distance to goal, a regularizer on joint torques and velocities, and a penalty for the tips slipping on the cube's surface. The last term encourages a firm grasp on the cube which is helpful for following the planned motion of our hand-designed controller.

While we planned to train models with domain randomization and real data, we did not have enough time. Instead, we use the policies trained during phase 1 and apply them directly to the real system. Surprisingly, despite substantial domain shift and changes to the base policies, we find the phase 1 policies trained with our hand-designed motion planning controller are able to improve performance on the real system. We expect further improvements by applying sim-to-real methods or fine-tuning with real data. For further discussion on these points, see the future work section.

Algorithm	Level 1	Level 2	Level 3	Level 4
Force Control (FC)	−10051 ± 5493	−28817 ± 4773	−28414 ± 8957	−45320 ± 5654
FC+Residual Learning (RL)	−11192 ± 6768	−36405 ± 7667	−25324 ± 11920	−45370 ± 13388
FC+Motion Planning (MP)	− 7973 ± 7898	−13559 ± 9597	−13842 ± 12688	−23517 ± 10680
FC+MP+Tip Adjust. (TP)	**−2673 ± 1166**	−8115 ± 8443	−4830 ± 5005	−32572 ± 15701
FC+MP+RL	−8018 ± 8011	−7167 ± 2170	−8812 ± 2996	**−20448 ± 6154**
FC+MP+TP+RL	−3330 ± 4331	**−2877 ± 204**	**−3893 ± 2453**	−22301 ± 11443

Table 2: Mean and standard deviation of rewards for each algorithm considered. For each algorithm we submitted 10 jobs and compute rewards from downloaded logs. These are not official results, only what we observed by running our own evaluations. Tip Adjustment refers to the method described in Section 1.3 which adjusts the planned motion to account for slippage between the cube and fingertips. The video of each level is available here.

2 Results

We test both of our hand-designed policies with and without residual policy learning by running 10 episodes for each level on the real system. The performance of each algorithm is shown in Table 2. Since we used a similar approach in phase 1, we show the simulated performance from phase 1 in Table 1 for comparison.

Comparing results from both phases, we find that our force control policy performs substantially worse on the real system. This is because it is very sensitive to noise in the cube's pose estimate. Since the cube's exact position and contact normals are used in determining what tip force to apply, position and orientation errors can cause sporadic forces to be applied to the cube. This often results in the cube being dropped.

Additionally, our motion planning policy assumes that there is no slippage between the cube and fingertips. This is often not the case on the real system. We find that adjusting the finger tips to drive the cube to the goal after the execution of the plan to be an effective solution to this problem (as shown in Table 2). However for level 4, the selected grasp of the cube may only be feasible near the goal location but not at the exact goal. In these cases, driving the cube to the goal location can result in a dropped cube.

As we mentioned in Section 1.5, we naively use a residual policy that is trained for phase 1 to get the results in Table 2. Given the non-negligible domain shift between the simulation and real environment, it is unexpected that the residual policy will transfer. However, we observe that these policies substantially improve the performance of the motion planning policy. We hypothesize that the motion planning policy helps transfer by constraining motion to be close to the predefined plan and that within these constraints the residual policy helps maintain the grasp on the cube. Residual policies trained with the force control policy do not have this property and fail to generalize.

3 Ongoing and Future Work

Due to time constraints we were unable to implement everything we had planned for phase 2. Specifically, our approach described above does nothing to help learned policies cross the sim to real gap. We are currently working on training policies with domain randomization. Domain randomization [3, 6, 7] aids transfer by forcing the learned policy to be robust to different dynamics properties, such as friction and mass, by randomizing the parameters across episodes. Additionally, the policy can be made robust to unmodeled effects by applying random disturbances and adding noise to the observations and actions. We expect that models learned with domain randomization, and other strategies such as fine-tuning on real data, will improve our results.

Our approach using residual policy learning relies on our ability to design a decent controller for the task. This may be possible when manipulating a cube, but can become much more challenging for other tasks such as picking up and writing with a pen. In this case an end-to-end learning approach that leverages demonstrations may be more successful. Learning from demonstrations has proven an effective method for robot learning systems. To this end, we are investigating ways to collect simulated demonstrations by tracking finger or hand movements. We plan on then leveraging those demonstrations during training through techniques such as behavioral cloning.

References

[1] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2019.

[2] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[3] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[5] Tom Silver, Kelsey R. Allen, Joshua B. Tenenbaum, and Leslie Pack Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[6] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2018.

[7] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[8] Manuel Wüthrich, Felix Widmaier, Felix Grimminger, Joel Akpo, Shruti Joshi, Vaibhav Agrawal, Bilal Hammoud, Majid Khadiv, Miroslav Bogdanovic, Vincent Berenz, et al. Trifinger: An open-source robot for learning dexterity. *arXiv preprint arXiv:2008.03596*, 2020.