Real Robot Challenge Phase 2

ThriftySnipe

Abstract

In our attempt to solve Phase 2 of the Real Robot Challenge, we reinforcement learn a control policy, in simulation, conditioned on delivering a single chosen dice to a single chosen goal. To further simplify the problem faced by the policy, we provide it with privileged state information from the simulator. Upon transfer to the real robot, an image processing technique is employed to estimate this state information. Ultimately, Phase 2 proves significantly more difficult than Phase 1 and our method struggles to solve the task, both in simulation or indeed reality.

1 Rearrange Dice

The goal of Phase 2 of the Real Robot Challenge (RRC) is to use the TriFinger robotic platform [1] to rearrange 25 dice into a target pattern. Images of the robotic arena are provided from three different camera angles at 10 fps. Unlike Phase 1, no estimates of object positions or orientations are provided, however, a segmentation function is available which produces a binary mask of cubes in the input image.

2 Method

2.1 Training the Control Policy

We train a control policy with Deep Deterministic Policy Gradients (DDPG) [2] and Hindsight Experience Replay [3] in a domain randomized [4] simulated environment[1]. To simplify the task faced by the policy, it is trained to move a single chosen dice (d_{chosen}) to a single chosen goal position (g_{chosen}). We will refer to this as a 'focused' policy. The dice which must be moved, and the goal position it must be moved to, are selected via a simple algorithm (see Algorithm 1). This algorithm chooses the dice-goal pair closest to each-other but at least 2cm apart (i.e. it chooses a dice that has not yet satisfied its closest goal). This choice is recalculated every 15 time-steps.

To further simplify its training, the policy is provided with ground-truth dice coordinate information only obtainable in simulation. Thus, the focused policy takes the following inputs: (i) the robot joint positions, velocities, and torques, (ii) the coordinates of d_{chosen} and g_{chosen}, (iii) the coordinates of the remaining dice in the environment. Dice orientations and velocities were omitted as they were found to be somewhat redundant and very difficult to obtain on the real robot. Pure torque control of the robot joints is used with an action frequency of 20 Hz.

Hindsight Experience Replay: The achieved goal is the position of the chosen dice: $ag = d_{chosen}$, and the goal is simply: $g = g_{chosen}$. The reward function used (see Equation 1) means the policy is only concerned with bringing d_{chosen} to g_{chosen}.

$$r = \begin{cases} 0 & \text{if} \quad \|ag - g\| \leq 2cm \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

Curriculum learning: In practice, we initially train the focused policy with just two dice in the arena, before iteratively increasing to a maximum of 25 dice. The number of inputs to the actor-critic neural networks are increased accordingly to accomodate the new dice in the observation vector. New weights are initialised to 0 to ensure behaviour is not initially affected by the new inputs [5].

[1] Our domain randomization implementation is equivalent to the one we used in Phase 1

Algorithm 1 Dice-goal selection algorithm

d_{chosen}, g_{chosen} = previous d_{chosen}, g_{chosen}
every 15 timesteps **do:**
    Obtain all dice positions and all goal positions.
    Calculate pairwise distances from each dice to each goal.
    Find closest goal to each dice, disregarding goals completed by other dice.
    d_{chosen}, g_{chosen} = closest resulting dice-goal pair where distance between is $>$ 2cm.
return d_{chosen}, g_{chosen}.

2.2 Detecting Dice Positions

Unlike in simulation, dices positions are not freely available on the real robot and thus had to be extracted from images. To do so, we first use the provided segmentation function to produce a binary mask of the dice from the original image. Then, we find the boundaries of each white region, compute its center, and project the center to the real world. These centers are assumed to be dice positions. We perform this operation for all three cameras angles, then return the centers from the angle that detects the most dice; this is to minimise the number of detected centers that are in fact multiple dice clumped together (see Figure 1).

To project the dice positions from the image plane to real world coordinates, we assume the task is solved via pushing of the dices around the arena (i.e., the dice are not be picked up). Therefore, knowing the dice size is 22 mm, we constrain the z coordinate to $z = 0.011$ and reduce the problem to the following equation:

$$\begin{bmatrix} X \\ Y \\ 0.011 \end{bmatrix} = \left(s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} A^{-1} - t \right) R^{-1} \tag{2}$$

where A corresponds to the intrinsic camera matrix, R and t are the rotation matrix and translation vector and s is the scaling factor. We refer to this method as *image2coords*.

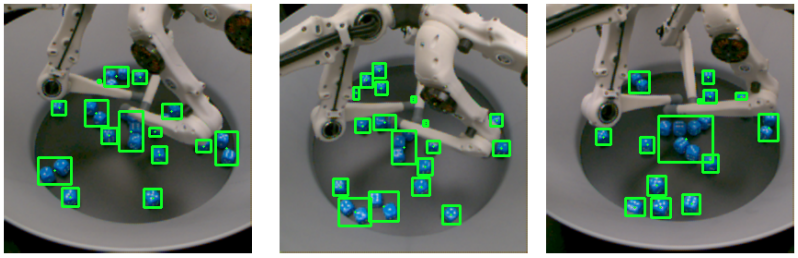

Figure 1: Dice detection using *image2coords* on the three camera perspectives of the real robot. The positions from the perspective which yields the most detections are used.

2.3 Real Robot Deployment

Our policy expects a fixed number of dice as input, however, the number of dice detected by *image2coords* can vary. To account for this, we conservatively use a version of the policy that only expects five dice positions as input, and perform the following steps: First, we select d_{chosen}, g_{chosen} based on the goals of the episode and the dice positions detected by *image2coords* (using Algorithm 1). Then, we select the closest four detected dice positions to d_{chosen}. These four positions are input along with d_{chosen} to satisfy the policies requirement for five dice as input.

3 Results

3.1 Simulation

In simulation, receiving true dice positions as input, our focused policy can deliver d_{chosen} to g_{chosen} at a $\sim$90% success rate. However, in doing so it carelessly disrupts other dice it has moved to other goals. This behaviour occurs because our reward function is oversimplified and does not punish such disruptions. Thus, the policy makes little progress towards rearranging all 25 dice into the desired pattern. As we decrease the number of dice in the arena, and the arena becomes

less crowded, performance improves. Yet, this number must be decreased to below five before the desired pattern can be matched reliably.

3.2 Real Robot

With our policy struggling simulation, producing any visibly meaningful behaviours on the real robot proves beyond its capabilities. It attempts to move the dice to goals but fails to gain any real control over them. Generally, most dice end up stuck around the circumference of the arena. The deficit in performance on the real robot is due to (i) the domain gap (e.g. the dice slide around much more freely in the real arena), and (ii) inaccuracies and irregularities in the estimated dice positions.

4 Discussion

The *Rearrange Dice* task is an excellent challenge that we found to be significantly more difficult to solve with reinforcement learning than Phase 1's *Move Cube on Trajectory*. The potential reasons for this are now briefly discussed.

Firstly, *Rearrange Dice* requires 25 dice to be moved to 25 goals, a significantly increase in complexity versus the single cube scenario found in *Move Cube on Trajectory*. Before turning to our 'focused policy' approach, we attempted to reinforcement learn a completely unstructured policy. These brief attempts involved HER, and variations such as 'multi-criteria HER' [6] to handle the multi-criteria nature of *Rearrange Dice*. However, even with two dice and two corresponding goals, these unstructured methods performed significantly worse than in the single dice setting.

Secondly, *Rearrange Dice* does not come with estimates of the dice positions. Thus, we must either: (i) estimate the dice positions ourselves, or (ii) learn the control policy directly from images. For the most part, we went down the route of (i), but ultimately struggled to obtain accurate position estimates. Option (ii) was mostly neglected as we had yet to learn a robust policy even with perfect state information. We did train an autoencoder to extract features from the segementation images, but our brief attempts to use these features to train an image-based policy were unsuccessful.

5 Future Work

Our attempts to solve this task present several potential routes for future work. Firstly, there is room for an improved multi-criteria version of HER which can better tackle the complicated goal-space of *Rearrange Dice*. Secondly, we note that our dice-goal selection algorithm for the focused policy could be seen as a manually implemented attention mechanism. Advances in learnable attention [7] may be useful in any future attempts to learn *Rearrange Dice* from scratch with an unstructured policy. Thirdly, there is room for better use of deep learning to obtain improved estimates of the dice positions. Among other attempts, we trained a ResNet-18 [8] to do so but results were unsatisfactory. Finally, an investigation into the possibility of learning a control policy directly from images in the simpler *Move Cube on Trajectory* could yield interesting results.

References

[1] Wüthrich, Manuel, et al. "Trifinger: An open-source robot for learning dexterity." arXiv preprint arXiv:2008.03596 (2020).

[2] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).

[3] Andrychowicz, Marcin, et al. "Hindsight experience replay." arXiv preprint arXiv:1707.01495 (2017).

[4] Tobin, Josh, et al. "Domain randomization for transferring deep neural networks from simulation to the real world." 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2017.

[5] Manela, Binyamin, and Armin Biess. "Curriculum Learning with Hindsight Experience Replay for Sequential Object Manipulation Tasks." arXiv preprint arXiv:2008.09377 (2020).

[6] Lanier, John Banister. Curiosity-driven multi-criteria hindsight experience replay. University of California, Irvine, 2019.

[7] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

[8] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Acknowledgments

This publication has emanated from research supported by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund and by Science Foundation Ireland Future Research Leaders Award (17/FRL/4832).