Assignment 6 - Integrated Grasp

CMPSCI 603: Robotics

Sanuj Bhatia

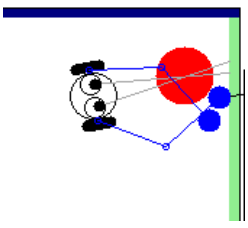
May 9, 2018

1. **Description of penalty/rewards**

   A fixed penalty of -5 is given per action. This penalty is applied to the reward tally every time the state changes, and is technically applied because of the last action executed (but since all actions have an equal penalty, it is merely a technicality). As for the rewards, only the reciprocal of the wrench residual was first added to the reward tally, and that too in case a correct and updated ϕ_w exists, which happens when FClosure() != NO_REFERENCE and the last action executed was actually FClosure(). This was done so that executing a SEARCHTRACK when the ball is in front of Roger will prevent the addition of a reward from the wrench residual - executing the correct action should lead to the correct reward.

   Using this strategy, Roger was able to learn that grasping the ball well is desirable when the ball is in front of him - but he did not learn how to search for or approach the ball and get into a good, stable position for grasping it. I think this was because the reward only gives incentives to achieve the goal, but does not direct Roger towards achieving it. Using various values of α and γ, Roger would always learn to grasp but not to successfully search and chase the ball.

   To help him with this, **I added rewards** for reducing distance to the ball, and for keeping the ball tracked well with both eyes. Whenever Roger executed actions like Chase or ChaseTouch, he would get a reward $\frac{0.6}{d}$ where d is the distance to the ball - the maximum reward is $\frac{0.6}{0.5} = 1.2$, achieved when Chase (or ChaseTouch) converges to a comfortable distance away from the ball (0.5m). The tracking reward is given when executing Track or SearchTrack, and has a value of $\frac{1.0}{0.1+e}$ where e is the angle of the base to the ball - a direct gaze has a maximum reward of 10. Note that this reward is not granted when executing FClosure, Chase, or Touch, even though they call stereo observation to track the ball. This is because the purpose is not to simply encourage tracking behavior, but to encourage actions that help Roger achieve the end goal. The numerator for both these rewards was decided after a number of (unsuccessful) trials, where Roger would begin to prefer Chase because the rewards added up to a significant number, and would ignore searchtrack. Observing the actions and rewards on the console gave me hints about increasing or decreasing the numerator, for the lack of an empirical method to decide on them. The wrench residual reward thus became $\frac{3.0}{\phi_w}$, dominating over the other rewards always. The ball distance reward has a very small numerator to reduce the possibility of ***fake rewards*** as well, an example of which is shown in Figure 1. In this position, Roger will probably never be able to grasp the ball. The ball is in a position where the distance is small and the reward from the distance metric accumulates, but only if the numerator is large.



These rewards make up a reward function that is nowhere close to perfect, and some points to consider will be discussed in (3). A reward for keeping his arms in the home position when not in a position to grasp the ball could potentially lead to a much better learning experience for Roger. I say this because if Roger stretches out to touch the ball but ends up punching it away, then his blue arms are in the way and his vision is severely hampered by the obstruction. The accuracy of estimating the position of the ball reduces, along with the fact that outstretched hands are a bad position to begin grasping the ball, since its approach is now blocked as shown in Figure 2. To include this reward, we would need to include the HomePosition() function call to Search, Track, Chase, and SearchTrack (this behavior is currently only exhibited in ChaseTouch). Since these functions are already well integrated and tweaked for the RL problem, adding this home position function to them led to various unseen setpoint variations in the arms (since Roger executed actions randomly in explore mode) and did not let him obtain a correct grasp. Regardless, it is worth thinking about. Any reward we include is a type of heuristic for Roger to know what behaviour is better in certain situations, and since we are looking for a specific sequence of actions (each of which we would wish to see to its convergence before moving onto the next) that lead to the goal, rewards that hint at correct behavior could lead a much better policy.

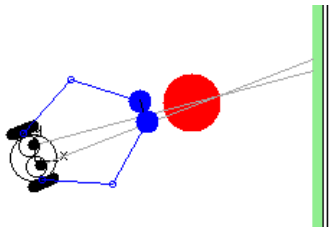
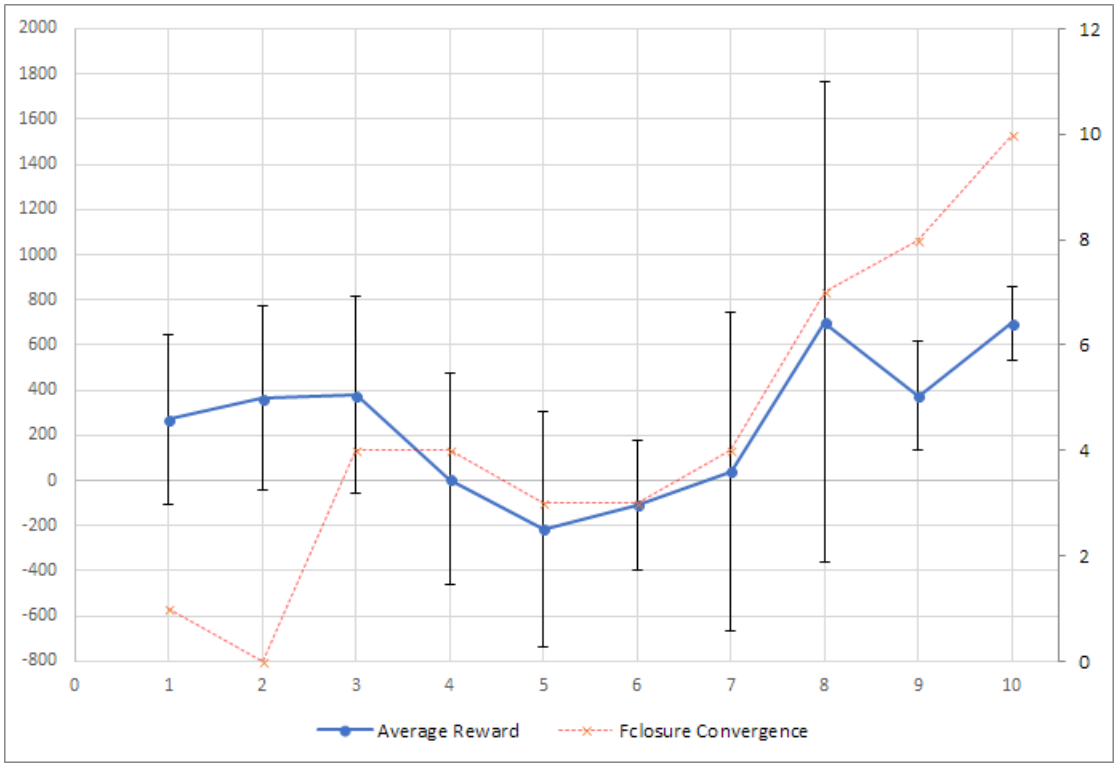


Figure 1: The ball is barely visible and as it approaches Roger, it will be punched away. If instead Roger's arms were in the home position, the balls's approach would lead to a smooth beginning of a grasping motion

2. **Performance Curve:** Average reward and convergence of FClosure out of 10 trials (FClosure + Macros only).



3. **Discussion**

The plot above shows in blue the average (and standard deviation - the variance values were too large to depict in a chart) of the sum of rewards obtained in each of the 10 trials until either the convergence of FClosure() or until 200 state transitions, and therefore 200 different actions, have been picked, whichever comes earlier. The red line with its axis on the right shows the number of times out of 10 that the trial ended because of the convergence of FClosure().

Roger was trained in 4000 iterations, with the policies being saved every 200 iterations. I picked every second policy and ran 10 trials with Roger acting according to that particular policy. The ball was gently put against a wall and Roger always found the ball moving with a small velocity - this is done to make it harder for Roger to control, but not to make him chase the ball all over the room. There are only 27 states in this problem, with 3 action choices for every state. For training, I used a linearly increasing ϵ value beginning from 0.2 at iteration 1 and going to 0.8 at 4000. Before beginning an iteration, I generate a random number, and if it comes less than ϵ I use the explore mode and pick actions with a greedy policy, and use the exploit mode otherwise, picking actions randomly.

In the first three policies, Roger does not learn to chase the ball, and so only gets to attempt a grasp if the ball goes through his reachable workspace. For policies 4-7, it is observed that Roger searches for and chases the ball successfully, but tries to attempt FClosure() too soon - seemingly before ChaseTouch has converged. This makes it so that the ball is still moving and Roger tries to slowly grasp it, but it escapes through his ethereal arms. As the ball moves away from Roger, a lot of state transitions happen and penalties of -5 accumulate fast. This is interesting, as it is very close to the desired behaviour of the reward system - the sum of rewards can be seen increasing as the ball comes closer to Roger and he grasps it, and then decreasing as he fails to form a stable grasp and punches it away. From policy 7 onward, Roger begins to learn to try and touch the ball before he goes on to forming a grasp, and this touch helps in controlling the ball's motion and possibly being it to a stop for a stable grasp. The variance can be seen to decrease as Roger learns this more reliable way to grasp the ball, and gets 10/10 balls in the last policy, including some impressive swings to stop the ball mid-movement while rotating his base and controlling the ball perfectly to form a quick force closure. For this reason, the reward values for policy 10 aren't too high or spread out, since the robot tends to grasp the ball quickly and precisely with no time for rewards to accumulate.

There are some problems with the way the rewards are defined. First, the distance reward does not take into account if the ball moved closer to Roger because of Roger's actions or because it bounced off the wall at a certain angle and is now headed towards the robot. Since Roger has no awareness of the walls, the balls motion, and the interaction between the ball and the wall, everything that Roger perceives and receives a reward for is assumed to be due to his own actions. This makes the first few policies, where he only knows how to track the ball, accumulate high rewards because the ball is in vision and is bouncing off the walls towards or away from him.

When trying to train the Robot with FClosure() + primitives, there is a problem in the learning process where actions picked randomly try to counter each others setpoints. For example, when the robot tries to track the ball and foveate on it, search keeps getting called in between and set the eyes to 0, which ends up making Roger switch between the actual position of the ball and the 0 angle back and forth - although very slightly. Another example is of FClosure and Touch, where Roger is alternating between trying to touch the centre of the ball and form a stable grasp on it. One of the more destructive interferences is of running Chase and Search. Search does not care if the ball is in vision or not - it is converged when the base is at the heading it was set to. So, while chasing the ball, it will randomly rotate Roger and lose sight of the ball. Due to this reason, the macros we have built are much more useful and simply ChaseTouch (which runs searchtrack as well internally) and FClosure is probably good enough to make the robot learn a grasp.

This course was a huge learning experience for me (and for Roger), and I would like to take this space to thank you, Sir, for structuring it so well and putting in the work to get us to this level of robot building - all while making it fun. If you have plans to port the simulator to Java/Python in the future, I would absolutely love to help!