2R Manipulator with Reinforcement Learning

Giang Dao

April 9, 2019

In robotics, people try to convert complex and high dimensional information to describe in lower dimensional context where the information is collected from expensive sensors. After transforming the information, a problem is solved the problem with a very complex math. This method high-to-low dimensional transformation can lead to a data loss problem. This method also brings a problem of scalability in which a slightly different problem might require researcher manually solve and setup a machine to work differently. There is another difficulty in robotics that is how to describe an object looks like in robotics term.

In this project, I will create a simulation of a 2R manipulator with 3 controllable joints (2 links and 1 gripper) to reach a certain goal with extract orientation. To get more challenge, the simulation produce image as input instead of using direct joints angle, arm links, as well as goal image. I hope to deliver a deep reinforcement learning method to solve a path generation for this simple problem. This learning method is an end-to-end method which is not required handcraft information or any complex math involved. The method will take input of images (RGB image for manipulator configuration and gray scale image for goal configuration) and output a three dimensional vector corresponding to each rotation joint.

In the simulation, the 2R manipulator can take up to 50 interactions to reach the goal configuration before the simulation resets to different random manipulator and goal configurations. Each interactions consists of controlling 3 joints which is constrained to range only from $[\frac{-\pi}{18}, \frac{\pi}{18}]$. Each interaction will receive a reward of -1 except for the interaction that reaches goal configuration receives a reward of 0. Figure 1 shows an example of the simulation.

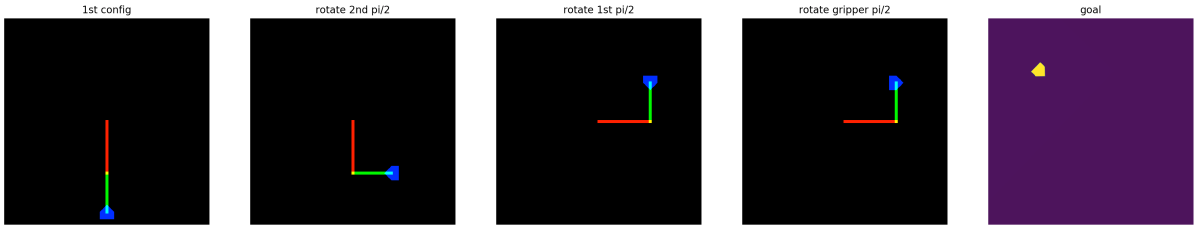

Figure 1: Movement samples of the 2R manipulator simulation with a sample goal configuration.

Q: What will accuracy of end-effector configuration be? Will it depend on image?
A: The accuracy is given from the simulated environment running behind the scene. Currently I'm setting first link as 160 units and second link as 120 units. The accuracy is determine if the end-effector is within 10 units radius of the goal configuration and the angle different maximum 10 degree. The learning agent can only access images information and nothing more.

Q: So will solve both inverse kinematics and trajectory generation?
A: I would think of it like what is the best action to take given current robot configuration and a goal configuration (also rule of maximum rotation angle apply). This algorithm use neural networks as function approximator to try to approximate all configuration and action value (Q-value) through the learning phase. Given a configuration, agent takes action that yields highest Q-value.

Q: Joints limit are [-10, 10]?
A: Joints limit are still 360 degree, but each rotation (action) angle limit is [-10, 10]. You can think of it like the maximum rotation can be taken in a short duration. To make the problem simpler, I did not contain velocity and acceleration in this project.

Q: What information does the vector contain?
A: 3D vector is representing 3 rotation angles of 3 joints. For example joint 1 takes the first value of the vector for rotation, joint 2 takes second value, etc.

Q: Can this extend to a 3R, 4R, 6R planar manipulator? What about spatial manipulator?
A: I believe the algorithm can apply to any manipulator environment. It's just the matter of how the simulated environment formulated. I'm pretty sure the algorithm can solve with direct joint angles and links information inputs, but not quite sure about image input instead. That is the reason I want to try.

Overall, I use the knowledge of the course to create a simulated manipulator environment, and apply deep learning method to solve a problem of getting from one configuration to a goal location with correct orientation. Also, I don't think planning method is good because things don't always go as planned. Learning method is on the other hand, it learns to react probably to the environment.