Project “Deep RL Arm Manipulation”

Dmitry Gavrilenko

1 Reward Functions

If the robot touches the target object, it receives +1 reward and the episode ends. In the gripper base scenario, the episode ends and the reward is received only if the robot touches the object with its gripper base part.

If the robot touches the ground or the episode exceeds 100 steps, the robot receives -1 reward and the episode ends.

The robot continuously receives a small reward dependently on the distance change between the object and the gripper. The distance change is averaged with its previous value in order to filter out noise or decrease latency artifacts of the observation->control pipeline. The distance decrease results in a small positive reward, and vice-versa. The small reward plays the role of the heuristics in the search algorithm, enabling the robot quickly finds solution towards the goal.

The distance change reward is scaled by exp(-distGoal) to increase the importance of correct motions that occur near the target object.

Additionally, the robot receives -0.01 reward for every step to penalize stationary behavior.

Reward function is implemented as assignment operators to rewardHistory variable in `./gazebo/ArmPlugin.cpp` module.

2 Hyperparameters

The camera input resolution is 64x64 pixels. Therefore, INPUT_WIDTH and INPUT_HEIGHT are also set to 64 pixels.

In case of LOCKBASE = true, the robot has only two active joints. NUM_ACTIONS equals 4 to provide two actions per joint.

“RMSprop” optimizer with LEARNING_RATE 0.1f is borrowed from catch sample. It provides better accuracy than “Adam” with LEARNING_RATE 0.001f.

REPLAY_MEMORY with 1000 samples and BATCH_SIZE 32 were tuned up experimentally.

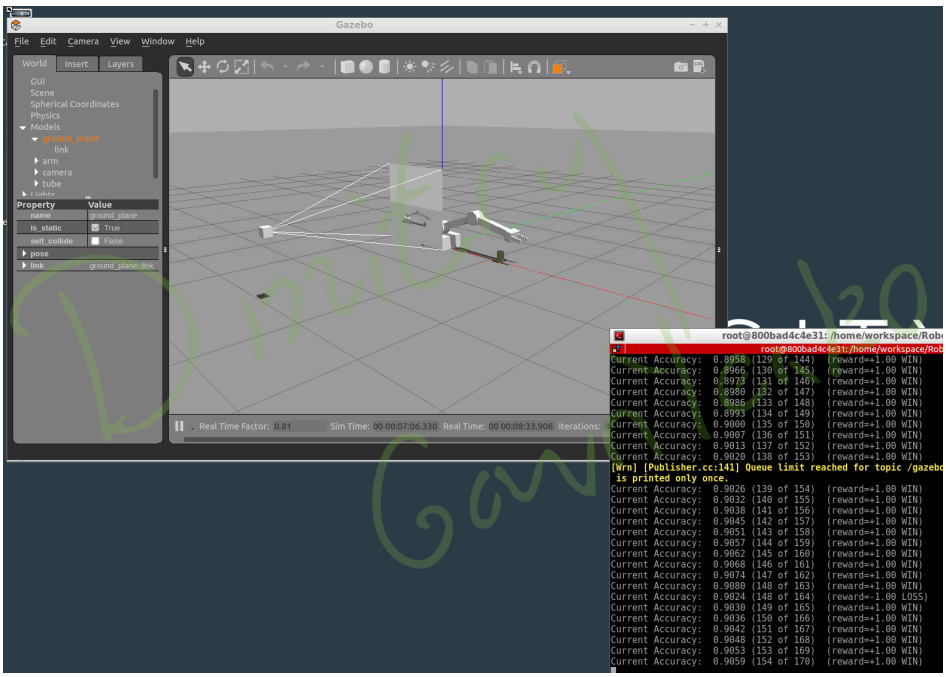
EPS_END is set 0.01 to prevent the robot unlearn good behavior. This decreases chances of random actions towards the end of the learning. The recommendation was borrowed from udacity-robotics.slack forum.

An experiment with LSTM and velocity controls showed good convergence rate but took a lot of training time (tens of thousands iterations). Therefore, to fit the training hardware resource constraints, the neural

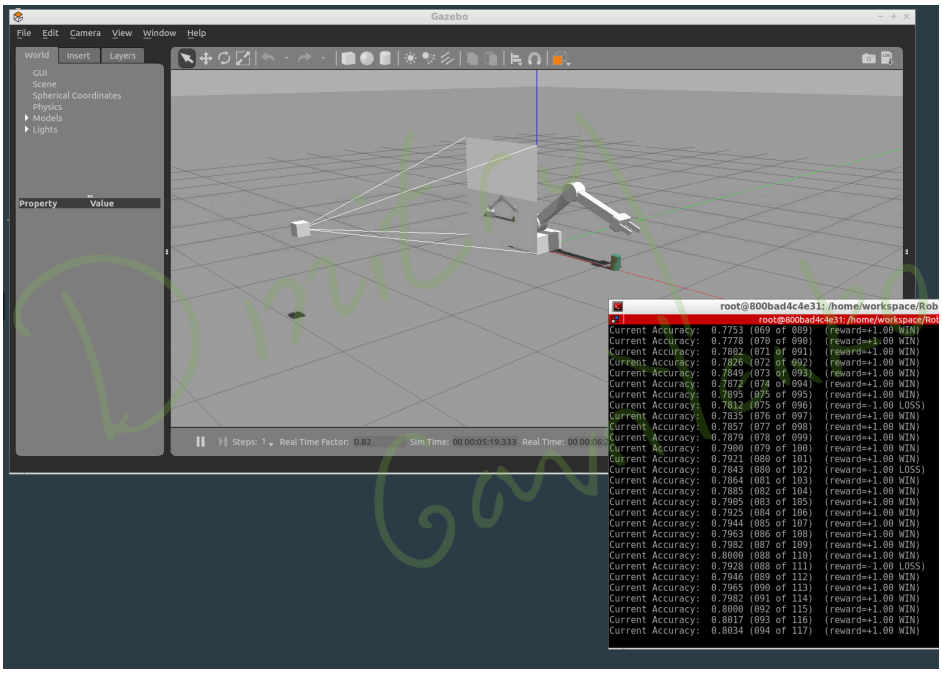
network does not use memory and controls the joint rotations directly. USE_LSTM is set false, and the training time takes hundreds of iterations.

3 Results

The robot reaches over 90% accuracy for the arm touching the object task in about 170 iterations, as shown in the pictures below:



The robot reaches over 80% accuracy for the gripper base touching the object task in about 117 iterations, as shown in the picture below.



The behaviors of robots for both objectives are similar. It looks like if the training procedure were longer, the robot would be able to achieve over 90% of accuracy even in the gripper base case.

The results demonstrate fast convergence rate due to high LEARNING_RATE, simple feed-forward neural network architecture, Markovian nature of the problem (the robot state can be fully described by the static image from the camera), nearly optimal values of hyper-parameters and the intermediate distance change reward.

The intermediate distance change reward seems to be the main contributing factor to the high robot accuracy and fast convergence. Without it, the robot would unlikely reach the goal and provide positive reward examples to initialize the neural network.

With this intermediate reward, the algorithm reminds a classical inverse kinematics problem: if the robot gripper was always equipped with some object proximity sensor, the camera input and the deep reinforcement learning approach would be redundant. It would be sufficient to minimize the distance to the goal with respect to the robot joint angles, lying in some valid safe sub-space of parameters.

This reveals the main shortcoming of the deep reinforcement learning approach: it can be robustly applied only to relatively short-term planning tasks, in which the reward function can be guided by smooth intermediate heuristics.

In the challenge scenario, in which the robot has 3 degrees of freedom and the position of the target object is random, the robot is able to achieve 37% of accuracy after 500 iterations. To achieve high accuracy in this scenario, more training time is needed with some ideas from the next section.

4 Future Work

The following ideas may further improve the quality of the DRL:

- Pre-train the neural network with labeled data: a recommended action per a given frame, prepared by a human operator
- Try Temporal Difference learning
- Parallelize the training task against multiple simulated or real robots, working in parallel on different computers, and filling in replay memory buffer
- Force the robot joints to operate in some safe space of possible values to exclude unsafe collisions with the ground and other robot parts.
- Penalize for high velocity of the gripper near the target object, and for non smooth behavior
- In case of non-constant latency presence between the camera image input and joint commands, consider training recurrent network with long-short-term memory in an attempt to predict dynamics of the joints.
- Stop training at some point in order to prevent over-fitting
- Evaluate accuracy in a slightly different context (using an approach, similar to cross-validation)
- Use skopt library to automatically search for optical hyper-parameter values