Project “Deep RL Arm Manipulation”

Dmitry Gavrilenko

# 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 very 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.

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

# 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 0.01f 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.

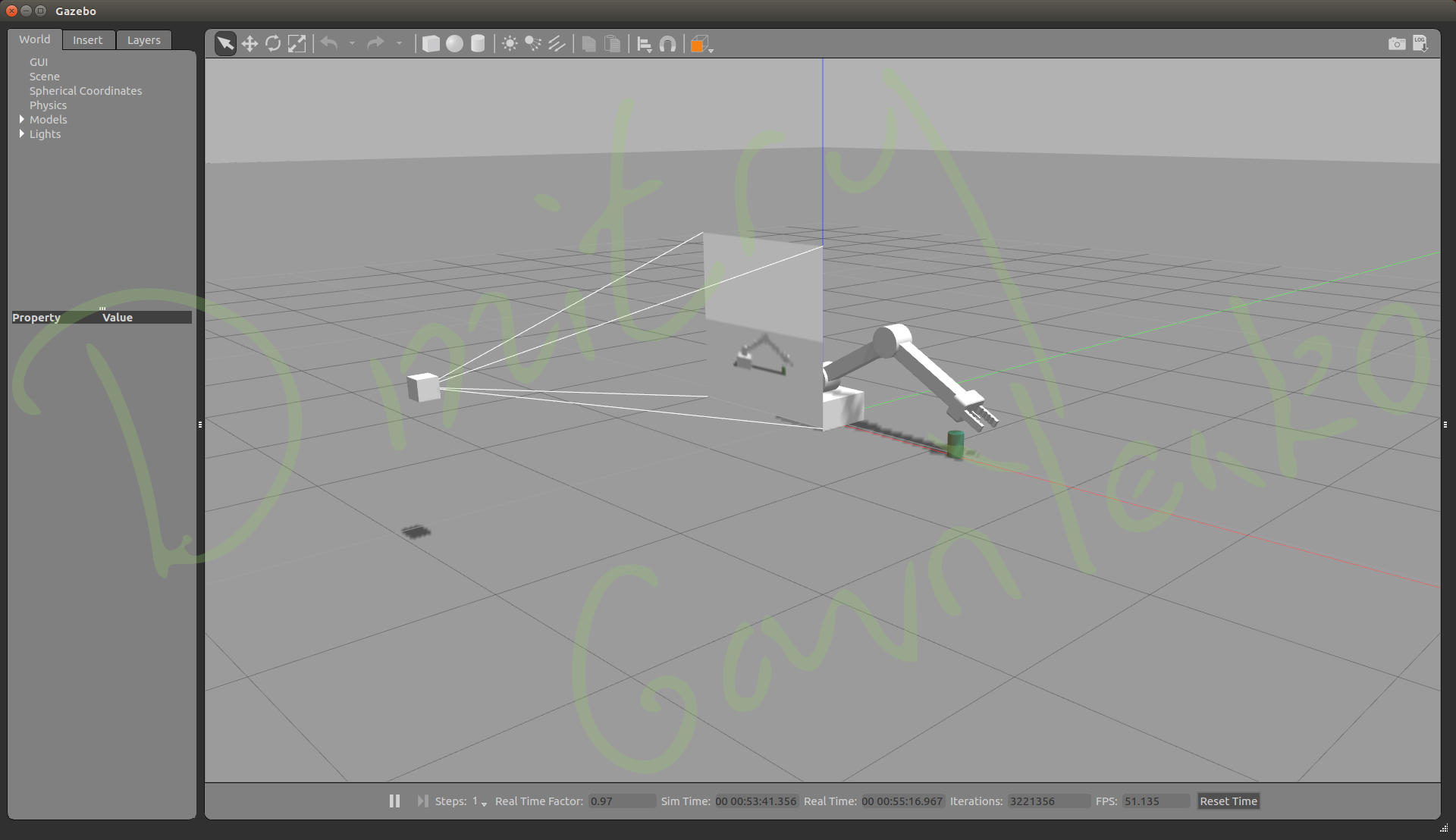
EPS\_END is set 0.01 to prevent the robot unlearn good behavior. This decreases chances of random actions towards the end of the learning. The recommendation was borrowed from udacity-robotics.slack forum.

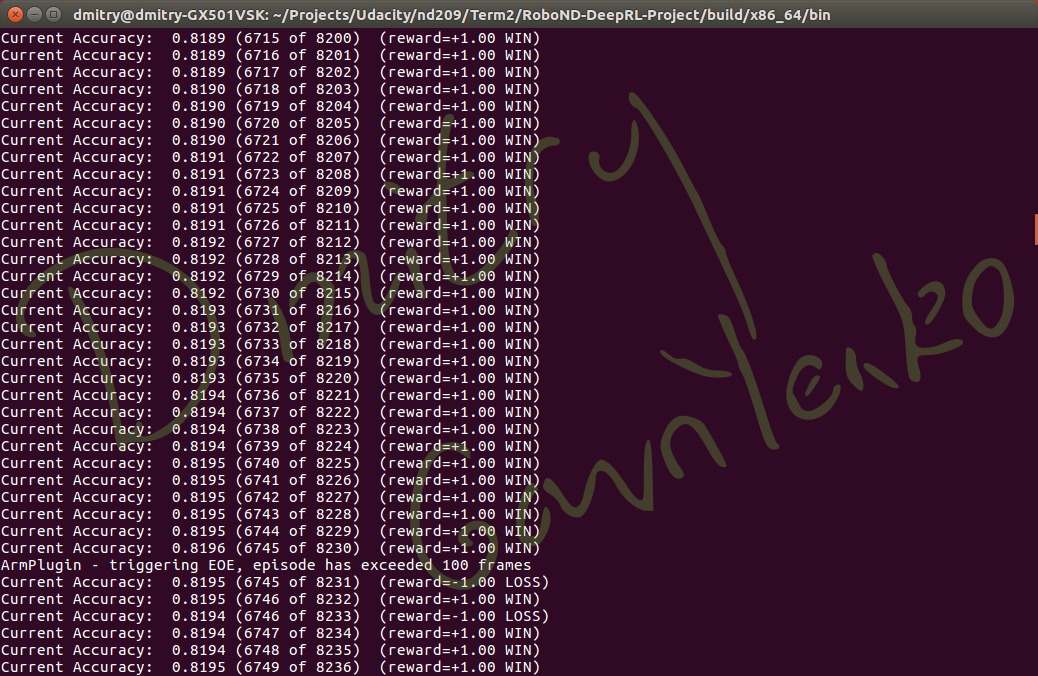
An experiment with LSTM and velocity controls showed good convergence rate but took a lot of training time (tens of thousands iterations). Therefore, to fit the training hardware resource constraints, the neural network does not use memory and controls the joint rotations directly. USE\_LSTM is set false, and the training time takes hundreds of iterations.

# Results

The robot reaches over 90% accuracy for the arm touching the object task in about 2000 iterations, as shown in the pictures below:

The robot reaches over 81.89% accuracy for the gripper base touching the object task in about 8000 iterations, as shown in the pictures below. Later, it starts decreasing its accuracy. Maybe, this problem is caused by over-fitting.





# 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