UC San Diego

Dexterous Manipulation with Deep Reinforcement Learning:Efficient, General, and Low-Cost

Presenter: Vijay Viswanath
21 May 2020

Outline

Motivation

study real-world model-free reinforcement learning as a means to learn complex dexterous hand manipulation behaviors

Answer the following questions:

1. Can model-free deep RL be practical in learning dexterous manipulation behaviors directly in the real world?
2. Can model-free deep RL algorithms learn on different hardware platforms and on different physical setups?
3. Accelerate the learning process using a small number of demonstrations
4. How do particular design choices of the reward function and actuation space affect learning?

Related Work

Robotic Grasping:

- Dexterous manipulation problem[2]: kinematics, force at contact points, formulating control problems.
- Training on simulation data: Dexnet

Model based learning:

- Estimate system dynamics and use reinforcement learning for learning policy[3]

Model free learning:

- Deep Reinforcement learning: Let a Deep NN learn to take observations and give out actions. The Deep NN learns the model through training.
- Use demonstrations and actual interactions to fill a replay buffer and train Deep Deterministic Policy Gradient [3]

Method outline

- Hardwares
- Task
- Algorithms

Hardware

Chosen to be robust to physical damage and cheap

Dynamixel Claw:

- 9 Degree of Freedom

Allegro Hand:

- 15 Degree of freedom

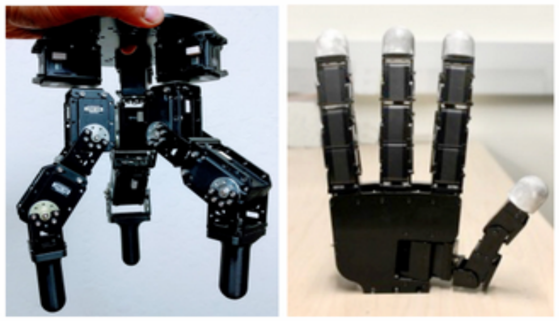


Fig. 2: Left: 3 finger Dynamixel claw. Right: 4 finger anthropomorphic Allegro hand

Task

Valve Rotation:

- turning a valve or faucet to a target position.
- Challenge: The fingers must cooperatively push and move out of the way
- Material of valve may change
- Contact points on valve is dynamic

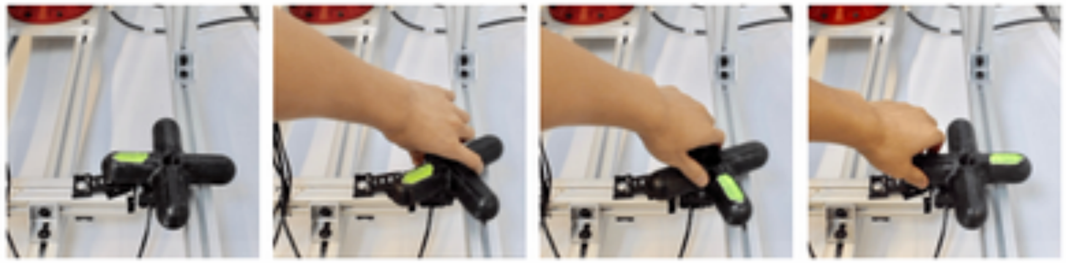


Fig. 3: Illustration of valve rotation

Task

Vertical Box flipping:

- rotating a rectangular box, which freely spins about its long axis
- Challenge: Need to learn alternating coordinated motions of the fingers such that while the top finger is pushing, the bottom two move out of the way.

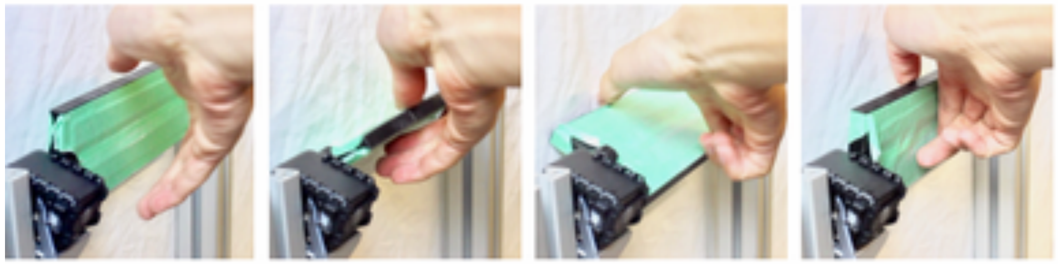


Fig. 4: Illustration of box flipping

Task

Door opening:

- Task require both Arm and hand. approach the door, grip the handle, and then pull backwards.
- Challenge: Larger degrees of freedom in possible actions given the additional arm, and the need to follow a sequence of actions

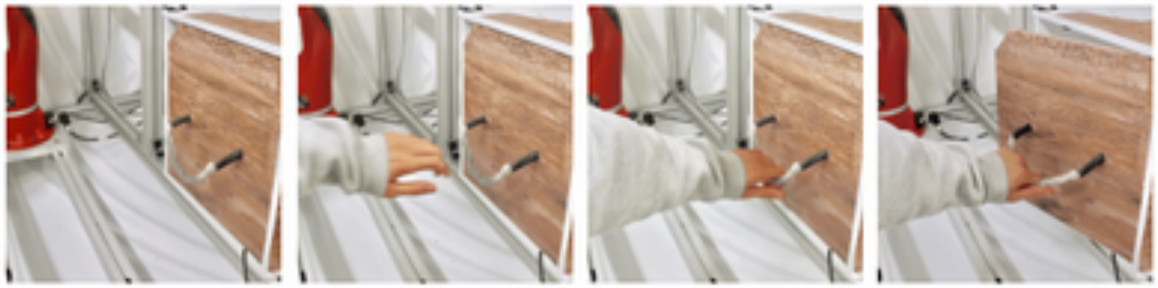


Fig. 5: Opening door with flexible handle

Tasks: Summary

Valve turning:

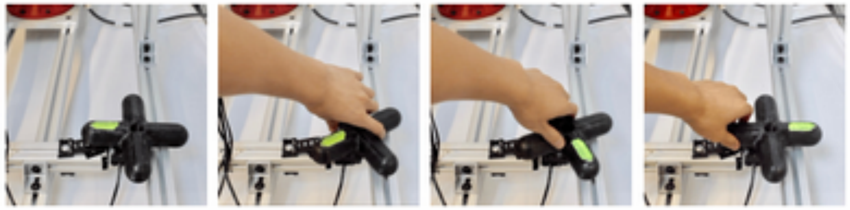


Fig. 3: Illustration of valve rotation

Box flipping:

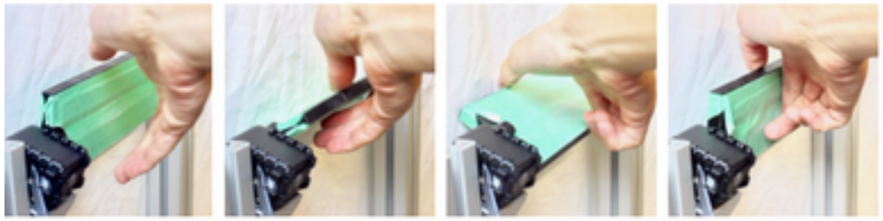


Fig. 4: Illustration of box flipping

Door opening:

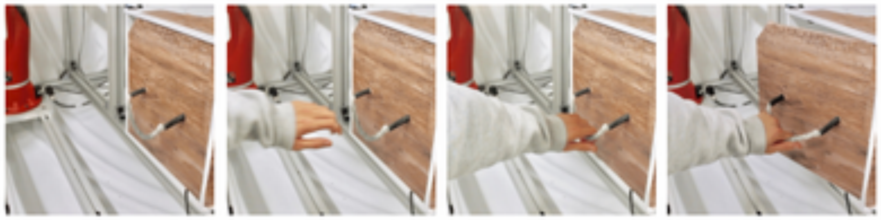


Fig. 5: Opening door with flexible handle

Algorithm

Considers the environment as a Markov decision process (MDP)

State Space: S

Action Space: A

Reward Function R: SxA -> real value

State Space here:

- All the joint angles of the hand
- Last action taken
- Task specific information

Environment for Valve turning

Task specific State Space here:

- current angle of rotation of the valve
- distance to the goal angle

Action Space:

- joint angles of the hand

Reward:

$$r = -|d\theta| + 10 * \mathbb{1}_{\{|d\theta|<0.1\}} + 50 * \mathbb{1}_{\{|d\theta|<0.05\}}$$
$$d\theta := \theta_{\text{valve}} - \theta_{\text{goal}}$$

Environment for Box flipping

Task specific State Space here:

- current angle of rotation of the box
- distance to the goal angle

Action Space:

- joint angles of the hand

Reward:

$$r = -|d\theta| + 10 * \mathbb{1}_{\{|d\theta|<0.1\}} + 50 * \mathbb{1}_{\{|d\theta|<0.05\}}$$

$$d\theta := \theta_{\text{box}} - \theta_{\text{goal}}$$

Environment for Door Opening

Task specific State Space here:

- Cartesian position of the arm,
- current angle of the door
- last action taken

Action Space:

- position space of the hand
- horizontal position of the wrist of the arm

Reward:

$$r = -(d\theta)^2 - (x_{\text{arm}} - x_{\text{door}})$$
$$d\theta := \theta_{\text{door}} - \theta_{\text{closed}}$$

Learning Policies

Performance of a policy: $\eta(\pi) = \mathbb{E}_{\tau \sim \mathcal{M}^{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right].$

Vanilla Gradient Ascend learning (REINFORCE):

$$\nabla_\theta \eta = \mathbb{E}_{\mathcal{M}^{\pi_\theta}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} r(s_t, a_t) \right] \quad (2)$$

stability and speed of learning improved by using the inverse of the Fisher Information Matrix

$$\mathcal{F}(\theta) = \mathbb{E}_{s,a}[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^T].$$

New update rule:

$$\theta_{i+1} = \theta_i + \alpha \mathcal{F}^{-1}(\theta_i) \nabla_{\theta_i} \eta$$

Natural Policy Gradient

Vanilla Gradient Ascend learning (REINFORCE)

$$g = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \hat{A}^\pi(s_t^i, a_t^i, t).$$

with Fisher Information Matrix

$$F_\theta = \frac{1}{NT} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \nabla_\theta \log \pi_\theta(a_t^i | s_t^i)^T,$$

Update Rule:

$$\theta_{k+1} = \theta_k + \sqrt{\frac{\delta}{g^T F_{\theta_k}^{-1} g}} \, F_{\theta_k}^{-1} g,$$

Use of demonstrations

Previous learning method referred as natural policy gradient (NPG)

With information from Human demonstrations, use demonstration augmented policy gradient (DAPG)

Demonstration dataset: $\mathcal{D} = \{(s_t^i, a_t^i, r_t^i)\}$

on-policy dataset obtained by rolling out π: $\mathcal{D}^\pi$

Demonstration Augmented Policy Gradient (DAPG) [5]

pre-train the policy π with behavior cloning on demonstration data D

Fine tuning using augmented policy gradient, with gradient:

$$g_{aug} = \sum_{(s,a)\in\mathcal{D}^{\pi}} \nabla_\theta \ln \pi_\theta(a|s) A^{\pi}(s,a) +$$

This term is from regular policy gradient

$$\sum_{(s,a)\in\mathcal{D}} \nabla_\theta \ln \pi_\theta(a|s) w(s,a).$$

Here w(s,a) is weighting function which controls how much close learning is to behavioral cloning

w(s,a) = 0 => Regular Policy gradient

w(s,a) = Large => Behavioral Cloning, that is learn to increase probability of choosing demonstrated action at states visited in demonstration

$$w(s,a) = \lambda_0 \lambda_1^k \max_{(s',a')\in\rho_\pi} A^{\pi}(s',a') \quad \forall (s,a)$$

Summary So Far

- Two Robotic hands

- 3 Tasks: Valve, box flipping and door opening

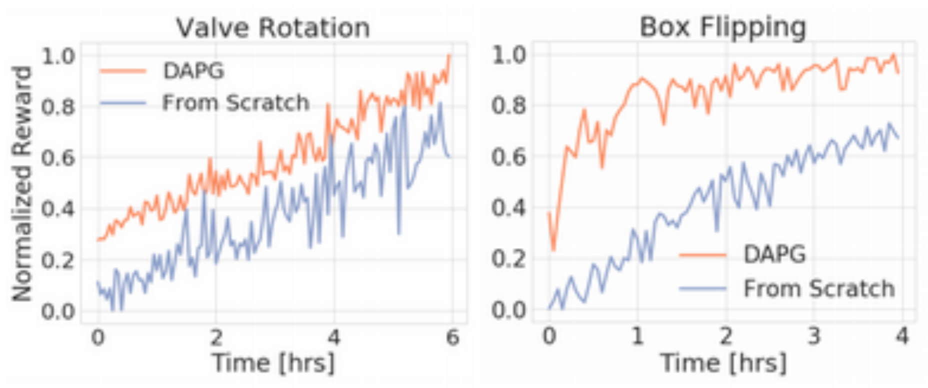
- Two learning policies for model free learning
  - Natural Policy Gradient
  - Demonstration Assisted Policy Gradient

Experimental Results

Task specific metrics defined for success (eg achieve > 30º for door in a trajectory)

Time taken to achieve 100% success rate:

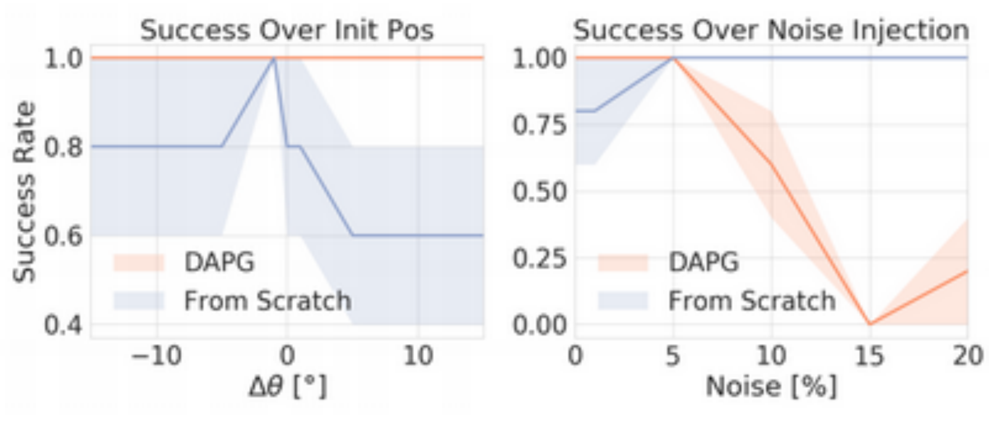
Task	From Scratch	DAPG
Valve	7.4	3.0
Box	4.1	1.5
Door	15.6	—



Learning progress with training from scratch

Robustness of learnt policies

What if start position is changed or noise added to observations?



DAPG is more robust with change in initial valve position,but is less robust with noise.

Hardware Pertubations

- Changing Robot:
  - Both Dclaw and Allegro hand learning took similar time to achieve 100% success rate on valve turning task

- Changing material:
  - Different valve material, such as soft foam
  - learns to focus the manipulation on the center of the valve where there is more rigidity.

Performance with Simulated Training

the randomization of PID parameters and friction was found to be crucial for effective transfer from simulation to real world

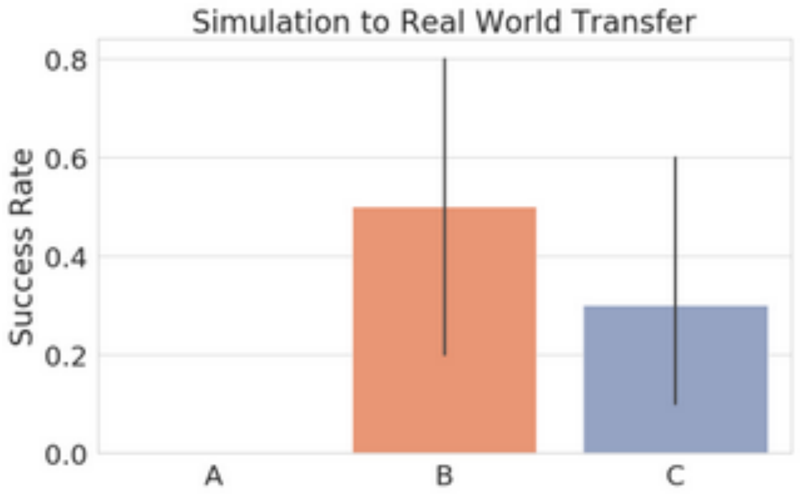
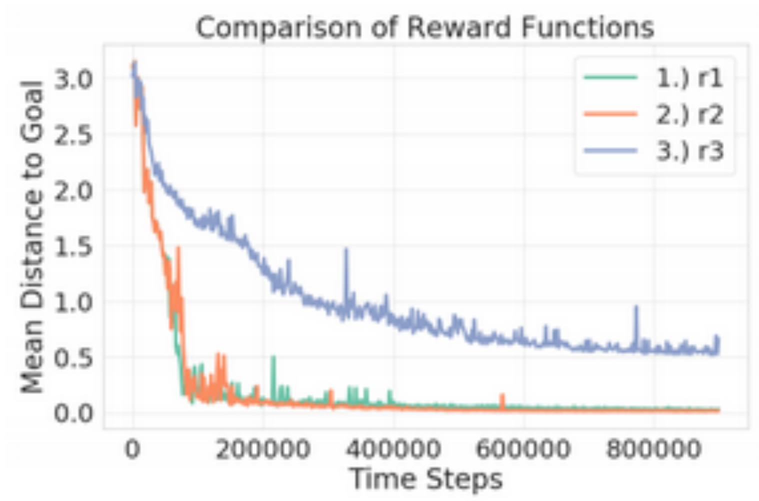


Fig. 19: Success rates using different sim2real transfer strategies for valve turning with Dclaw. A: No domain randomization. B: Randomization of position control PID parameters and friction. C: Same as B, but also including the previous action as part of the state space.

Impact of reward function

Reward 3: The control cost ensures smoother operation, but at the cost of efficiency in learning (less exploration)

1) $r_1 = -\|\theta - \theta_{goal}\|_2$
2) $r_2 = r_1 + 10 * \mathbb{1}_{\{r_1 < 0.1\}} + 50 * \mathbb{1}_{\{r_1 < 0.05\}}$
3) $r_3 = r_2 - \|v\|_2$



Questions ?

Conclusion:
Fast learning of Model free RL possible using real world data

Robust to devices and environmental factors

Demonstrations assist in faster learning

Simulation to real world training transfer require modeling random perturbations

References

1. Xhu, H., Gupta, A., Rajeswaran, A., Levine, S., and Kumar, V. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost.arXiv preprint arXiv:1810.06045, 2018
2. A. M. Okamura, N. Smaby, and M. R. Cutkosky, "An overview of dexterous manipulation," in Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, vol. 1, pp. 255–262, IEEE, 2000.
3. A. Gupta, C. Eppner, S. Levine, and P. Abbeel, "Learning dexterous manipulation for a soft robotic hand from human demonstrations,"
4. M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot,N. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards,"
5. A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," CoRR, vol. abs/1709.10087, 2017.