Vision Based Robot Manipulation Testbed for Reinforcement Learning

Department for Interdisciplinary Research
College of Engineering Trivandrum

July 9, 2020

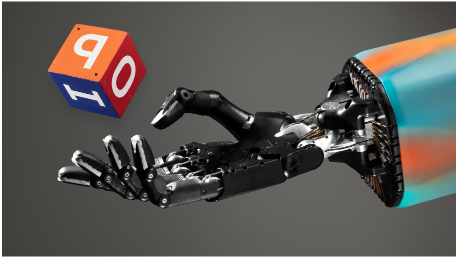
Guided by,
Linu Shine
Electronics and Comm. Engg.

Presented by,
Sreejith Krishnan R
Robotics & Automation

Motivation

Why robot manipulation?

- Assist humans in general tasks
- Build machines that can grasp and manipulate objects with human levels of dexterity



A reinforcement learning testbed provides,

- A standardized benchmarking environment for comparing performance of different RL algorithms
- An entry point for quickly testing RL algorithms for robot manipulation tasks enabling quick development
- A high performance framework for efficient training of RL algorithms

Literature survey I

Title	SURREAL: Open-Source Reinforcement Learning Framework and Robot Manipulation Benchmark [2] - Conference on Robot Learning - 2018
Methodology	• Open-source framework for benchmarking reinforcement learning algorithms • Decomposed architecture
Merits	• Allows scaling RL speed with computation power • Prebuilt standardized environments for common robot manipulation tasks
Demerits	• No integration with RayLib - A common framework for scalable reinforcement learning • No prebuilt support for experiment logging and tracking

Literature survey II

Title	Comparing Task Simplifications to Learn Closed-Loop Object Picking Using Deep Reinforcement Learning [3] - IEEE Robotics and Automation Letters - 2019
Methodology	• Uses autoencoder to encode high dimensional camera data to low dimensional encoding • Feed forward neural network predicts action from encoded data • RL to train neural networks
Merits	• No hand labeled data required
Demerits	• Low success rate (78%) for manipulation of objects in clutter by real robot • Non modular. Difficult to reuse model for similar task

Literature survey III

Title	Regularized Hierarchical Policies for Compositional Transfer in Robotics [4] - DeepMind - 2019
Methodology	Use hierarchical modular policies for continuous control.
Merits	• Best sample efficiency on both simulated and real robot • Uses MPO optimization algorithm which reduces the number of hyperparameters
Demerits	• High level tasks are not automatically decomposed to sub tasks • Low level policy is shared across all low level tasks making interpretability complicated • Transferring specific skills from sub-tasks policy in a predictable manner is difficult • Experiment results are obtained using model whose inputs include pose of objects in workspace

Research Gap and Objectives

Research Gap

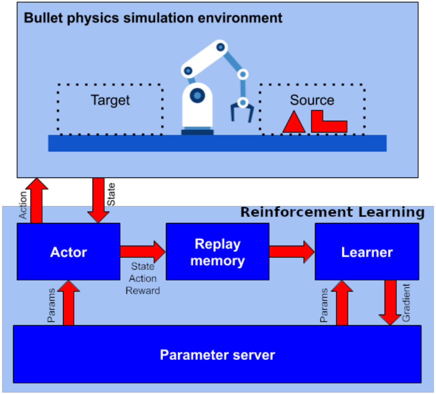
- Slow training data collection speed
- Non standard/readily available frameworks used

Objective

- Improve RL model training speed by running multiple simulations in parallel
- Use standard frameworks that support training distributed RL algorithms
- Prebuilt experiment logging and tracking
- Flexibility for adding different types of robot manipulation tasks like grasping, moving etc.

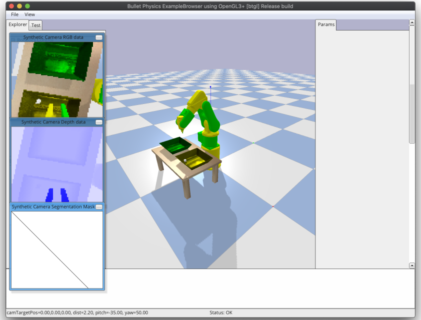
Methodology I

- Simulation environment created on Bullet Physics Simulator
- Experiments will be run in distributed fashion
- Simulator tuned for maximum throughput
- Integrated with RayLib for scaling reinforcement learning
- Experiment logging, tracking and visualization using comet.ml platform
- Easy to add new robot manipulation tasks



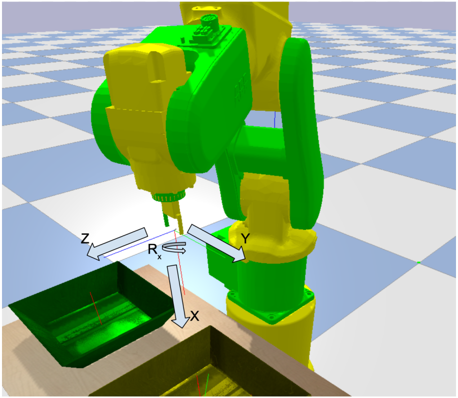
Methodology II

- Simulated robot tuned to have properties similar to real ABB IRB 120 robot
- Graphics rendering can be disabled to speedup simulator
- Grasp detection and collision detection algorithms for easy reward calculation
- Multiple simulator instances can be run parallel



Experiment Setup

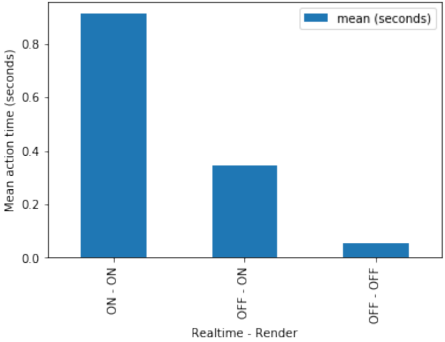
- Task is to move the object from yellow tray to green tray
- RGB-D camera is mouted on end effector
- Input to RL model is depth image from RGB-D camera
- Observation space is of shape 84x84x4
- Action space is $[\delta x, \delta y, \delta z, \delta r_x, open]$ which can be used for relative movement and rotation of end effector



Results

Simulator performance

- Mean action time of 0.053 seconds without rendering and 0.345 seconds with rendering
- 2.4 times faster than data collection from real robot when rendering is enabled
- 17.2 times faster than data collection from real robot when rendering is disabled



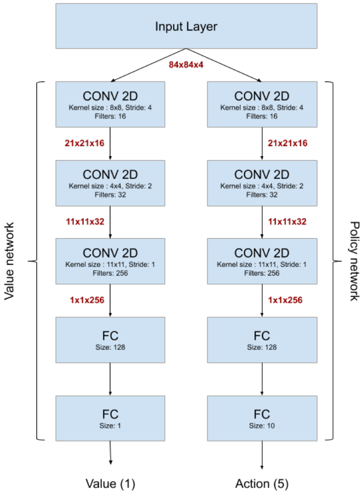
Render — Realtime	Mean	Std	25%	50%	75%
ON — ON	0.912	2.783	0.358	0.374	0.39
ON — OFF	0.345	0.583	0.198	0.206	0.214
OFF — OFF	0.053	0.023	0.046	0.047	0.048

Results

Baseline PPO

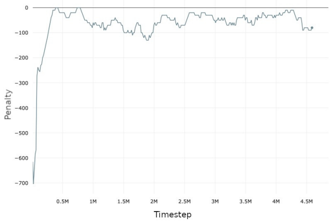
- Convolution layers are used for feature extraction
- Input in RGB-D 84x84x4 matrix
- Value network predicts the how good it is to be at a particular state
- Policy network directly predicts the mean and standard deviation of a Gaussian PDF from which actions are sampled for a particular state
- Train batch size is 10240 and SGD minibatch size is 512. Number of iterations per train batch is 30
- Data from an episode is added to train batch only after the it is complete

Architecture

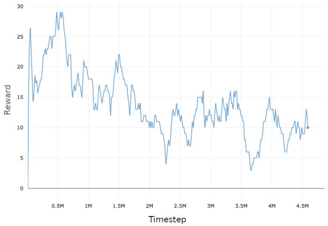


Results
Baseline PPO

- Model learns to avoid collision after training for 4M timesteps
- Optimum reward values for grasping and drop penalty is not reached after training for 4M timesteps
- From other research papers, PPO network needs to be trained for approximately 80M timesteps to reach optimum reward values

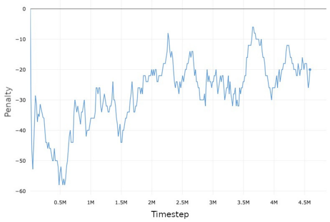


Collision penalty. Optimum value is 0

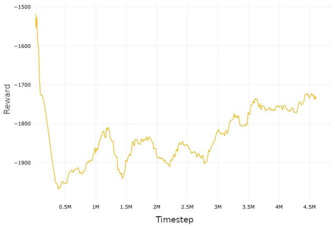


Grasp reward. Optimum value is 100

- Reward shaping is critical. Small changes in reward function can change the learning process.
- Providing +ve reward when end effector moves towards target and -ve reward when end effector moves away will not work
- Initializing each episode at a random state like grasped and not grasped can improve training speed



Drop penalty. Optimum value is 0



Mean reward. Optimum value > 200

Future scope

- Including DDPG and RHPO baseline models
- Evaluating baseline models on real robot
- Including more common robot manipulation tasks to testbed
- Including baseline support for multi agent robot manipulation tasks

Challenges I

- Initially objective of this project was to evaluate Regularized Hierarchical Policy Optimization (RHPO) for robot manipulation tasks
- To train RHPO for table clearing task, computations must be run on machine with 32 CPU cores and NVIDIA V100 GPU for approx 2 days
- In CET, computer cluster with more than 32 CPU cores is available. But it does not have GPU. Efforts were made to try and use compute cluster from CET for running simulations and use GPU from google colab for training neural networks, but slow networking speed made the training process extremely slow
- Also efforts were made to use both GPU and CPU from google cloud by using 300 USD trial provided by google cloud. But restrictions in running time of preemptible virtual machines prevented this strategy
- Hence the objective of the project was changed to development of testbed which require lower compute power since only simulator development is required.

Demo

References I

Richard Hodson.
How robots are grasping the art of gripping.
https://www.nature.com/articles/d41586-018-05093-1.
Accessed: 2019-09-24.

Linxi Fan, Yuke Zhu, Jiren Zhu, Zihua Liu, Orien Zeng, Anchit Gupta, Joan Creus-Costa, Silvio Savarese, and Li Fei-Fei.
Surreal: Open-source reinforcement learning framework and robot manipulation benchmark.
In *Conference on Robot Learning*, 2018.

M. Breyer, F. Furrer, T. Novkovic, R. Siegwart, and J. Nieto.
Comparing task simplifications to learn closed-loop object picking using deep reinforcement learning.
IEEE Robotics and Automation Letters, 4(2):1549–1556, April 2019.

References II

Markus Wulfmeier, Abbas Abdolmaleki, Roland Hafner, Jost Tobias Springenberg, Michael Neunert, Tim Hertweck, Thomas Lampe, Noah Siegel, Nicolas Heess, and Martin A. Riedmiller.
Regularized hierarchical policies for compositional transfer in robotics.
CoRR, abs/1906.11228, 2019.