In-Hand Robotic Manipulation via Deep Reinforcement Learning

Kapil D. Katyal
Johns Hopkins University
Applied Physics Lab
kapil.katyal@jhuapl.edu

Edward W. Staley
Johns Hopkins University
Applied Physics Lab
edward.staley@jhuapl.edu

Matthew S. Johannes
Johns Hopkins University
Applied Physics Lab
matthew.johannes@jhuapl.edu

I-Jeng Wang
Johns Hopkins University
Applied Physics Lab
i-jeng.wang@jhuapl.edu

Austin Reiter
Johns Hopkins University
Computer Science
areiter@cs.jhu.edu

Phillipe Burlina
Johns Hopkins University
Applied Physics Lab
phillipe.burlina@jhuapl.edu

Abstract

Deep learning (DL) has led to near or better than human performance in image classification or object/speech recognition. DL is now providing new tools to address autonomous robotic manipulation and navigation challenges. One of the fundamental capabilities necessary for robotic manipulation is the ability to reorient objects within the hand. In this paper, we describe an approach using Deep Reinforcement Learning (DRL) techniques to learn a policy to perform in-hand manipulation directly from raw image pixels. This paper presents an overview of the working prototype, the description of the algorithms and a working prototype using the Modular Prosthetic Limb (MPL) in a Gazebo simulation.

1 INTRODUCTION

Robust manipulation is a fundamental capability necessary for robotic systems to effectively interact with their environment. One of the key, and largely unsolved challenges in robotic manipulation is the ability to reorient an object within the hand as evidenced by the challenges in unmodeled dynamics, complex contact physics, errors in perception, and errors in actions. Several approaches to in-hand manipulation have been studied over the past several decades due to its relevance to real world applications as well as the numerous challenges associated with sensing, planning, and control. Michelman (1998) was one of the first to study in-hand manipulation by maintaining grasp stability during the manipulation task. This work involved force and position control through contact points throughout the trajectory and was evaluated using the Utah/MIT dexterous hand as the hardware platform. Odhner and Dollar (2015) studied precision manipulation and in hand manipulation utilizing under-actuated end-effectors. Han and Trinkle (1998) provide a formal mathematical framework for contact kinematics and rolling finger gaits for a three finger end-effector. Dafle et al. (2014) studied regrasping strategies to reorient objects within the hand. Levine et al. (2016) demonstrated great results using deep learning to perform hand eye-calibration. The work most similar to this effort was performed by Kumar et al. (2016). In this approach, Reinforcement Learning (RL) was used to...
learn a local linear model to control a pneumatic five finger hand by optimizing a linear Gaussian controller.

While all of these works produce significant results, they require some combination of precise models including manipulator kinematics, dynamics, interaction forces, high-fidelity tactile and/or joint position sensors available on-board the robot. The reliance on manipulator models makes transferring solutions to new robot configurations challenging and time consuming. This is particularly true when a model is difficult to define as in soft or under-actuated systems. In addition, requiring high fidelity tactile and position sensing becomes challenging in anthropomorphic hands given the space constraints. Traditional approaches use layers of piezoresistive films, optical arrays, or fluid based pressure sensors which can be costly, fragile or generally noisy. We believe model-based approaches that rely on high fidelity sensor information has resulted in algorithms that are too brittle to be applied in real world, unconstrained environments.

In this paper, we revisit high dexterity, in-hand manipulation with an emphasis on eliminating the reliance on precise models and on-board sensing. Our objective is to demonstrate the feasibility of performing in-hand manipulation using only image pixels from the scene. Specifically, our goal is to demonstrate that principles of deep reinforcement learning (DRL) (Mnih et al., 2013) can be used to develop a policy that maps image pixels to actions corresponding to a high rate of success when trying to rotate an object within the hand. Our approach is unique in that it makes no assumptions of the robot’s configuration, kinematic or dynamic model and has no dependency on tactile or absolute position sensing available on-board.

Our major contributions are 1) developing the first model-free, purely vision-based approach to performing in-hand manipulation that we are aware of; 2) developing tooling software that allows us to control a robotic system as well as capture image pixels and other state information from a Gazebo simulation environment; and 3) preliminary results demonstrating the potential of learning model-free in-hand manipulation.

2 METHOD

2.1 Deep Reinforcement Learning

RL has been widely used to allow agents (i.e. robots) to learn how to interact with the environment. Traditionally, RL can be formalized as a Markov Decision Process (MDP) which consists of an agent interacting with the environment. The agent selects actions that result in a reward causing a change to the environment as observed as the next state. Q-Learning is a form of RL where the primary objective is to develop a policy that allows the agent to select an action given the current state in order to maximize the expected reward. The Q-value correlates to the quality of choosing the action given the state and is iteratively updated by Eq. 1.

\[
Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha_t(s_t, a_t)[R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)]
\] (1)

where \(Q_{t+1}(s_t, a_t)\) represents the updated Q-value, \(Q_t(s_t, a_t)\) is the previous Q-value, \(\alpha_t(s_t, a_t)\) represents the learning rate, \(R_{t+1}\) is the immediate reward, \(\gamma\) represents the discount factor, and \(\max_a Q_t(s_{t+1}, a)\) represents the estimate of the optimal future value. In the simplest case, the Q-value is stored in a table with the rows and columns corresponding to the states and actions respectively. Using this definition of the Q-value, the selected action is defined by \(\pi(s) = \arg\max_a Q(s, a)\).

With Q-Learning, a compact representation of the system state is typically assumed for computational efficiency. In the case of the Atari 2600 game Breakout, the paddle position, ball position, and block locations could all be used to represent the state. For manipulation, the state of the system could be modeled as the Denavit-Hartenberg (DH) parameters defining the robot kinematics, the current joint angles and the location of the target object. The issue with traditional RL approaches is that the state is specific to the application and would not apply to other games or manipulators. The preferred approach would be to model the state using raw image pixels of the scene. This would allow maximum generalization; however, the challenge is the curse of dimensionality. Assuming the state is represented as 4 frames of an 84×84 pixel grayscale image, the state space is equal to \(2^{2564 \times 84 \times 84} \approx 10^{67970}\) dimensions. Clearly, the magnitude of the state space would make convergence infeasible using modern computing platforms. Instead of using a table to represent the
Q-values, a deep neural network (DNN) is used as a function approximator that maps raw image pixels to Q-values corresponding to potential actions. In the remaining sections, we describe the system architecture and the implementation details necessary to apply deep reinforcement learning to learn in-hand manipulation.

2.2 System Architecture

The system architecture, as described by Fig. 1b, consists of the deep reinforcement learning framework as well as the robot interface including a module able to bias joint angles of the robotic gripper and collect information from the simulation environment necessary to compute the reward based on selected actions.

2.3 Robotic Manipulator and Simulator

To perform in-hand manipulation, our goal was to evaluate the effectiveness of a high dexterity robotic hand. For this purpose, we leveraged the Modular Prosthetic Limb (MPL) developed by JHU/APL (Ravitz et al., 2013). The MPL consists of 17 actuated degrees of freedom (DOF) with 10 DOF in the hand. The MPL is also fully sensorized with position, force, vibration and contact sensing, however these sensors were not used as part of this experiment but are planned for future use as described in the conclusion.

Gazebo (Koenig and Howard, 2004) was used as the simulation environment which includes a high-fidelity simulation of the MPL. As part of this effort, we developed custom Gazebo plugins that allow our Deep Q-Network (DQN) to send control commands to the MPL, read state information (image pixels of the screen) and compute a reward based on properties of the objects in the simulation. The simulation environment consists of the MPL holding a cricket ball. The objective was to learn appropriate actions to take that would rotate the cricket ball along a specific axis while minimizing translation or rotation along the other axes. More details of this approach are provided in 2.4.

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Thumb Abduction/Adduction, positive angle delta</td>
</tr>
<tr>
<td>1</td>
<td>Thumb Abduction/Adduction, negative angle delta</td>
</tr>
<tr>
<td>2</td>
<td>Thumb Flex/Extend, positive angle delta</td>
</tr>
<tr>
<td>3</td>
<td>Thumb Flex/Extend, negative angle delta</td>
</tr>
<tr>
<td>4</td>
<td>Thumb Medial Joint, positive angle delta</td>
</tr>
<tr>
<td>5</td>
<td>Thumb Medial Joint, negative angle delta</td>
</tr>
<tr>
<td>6</td>
<td>Thumb Distal Joint, positive angle delta</td>
</tr>
<tr>
<td>7</td>
<td>Thumb Distal Joint, negative angle delta</td>
</tr>
</tbody>
</table>
2.4 DRL Framework

The DRL Framework uses raw image pixels as input from the Gazebo simulator and outputs the Q-value for each of the eight actions as summarized in Table 1 which emphasize motion of the 4 DOF thumb to perform rotation of the object. The neural network used to approximate the Q-value table is described in Table 2.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Input</th>
<th>Kernel Size</th>
<th>Stride</th>
<th>Num Filters</th>
<th>Activation</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>4 x 84 x 84</td>
<td>8 x 8</td>
<td>4</td>
<td>32</td>
<td>ReLU</td>
<td>20 x 20 x 32</td>
</tr>
<tr>
<td>conv2</td>
<td>20 x 20 x 32</td>
<td>4 x 4</td>
<td>2</td>
<td>64</td>
<td>ReLU</td>
<td>9 x 9 x 64</td>
</tr>
<tr>
<td>fc3</td>
<td>9 x 9 x 64</td>
<td>N/A</td>
<td>N/A</td>
<td>256</td>
<td>ReLU</td>
<td>256</td>
</tr>
<tr>
<td>fc4</td>
<td>256</td>
<td>N/A</td>
<td>N/A</td>
<td>18</td>
<td>ReLU</td>
<td>18</td>
</tr>
</tbody>
</table>

During training, the goal is to learn a Q-value for each action corresponding to the current state that represents a best estimate of the quality of selecting the given action. An $\epsilon$-greedy approach was used for random exploration with $\epsilon = 0.1$. A discount reward factor, $\gamma$ was set to 0.95. During the learning phase, the loss function described by Eq. 2 is used to update the weights of the DQN network after every episode consisting of 75 training steps.

$$L = \frac{1}{2} [R + \max_a(Q_{t+1}(s, a)) - (Q_{t+1}(s, a))]^2,$$

(2)

where $R + \max_a(Q_{t+1}(s, a))$ represents the target Q-value and $(Q_{t+1}(s, a))$ is the predicted Q-value.

The reward function is described by Eq. 3 which is a linear combination the linear and angular velocities of the cricket ball where $\beta_1 = \beta_2 = \beta_3 = -20.0, \beta_4 = \beta_5 = -10.0, and \beta_6 = +20.0$.

$$R = \beta_1|v_x| + \beta_2|v_y| + \beta_3|v_z| + \beta_4|\omega_x| + \beta_5|\omega_y| + \beta_6|\omega_z$$

(3)

These parameters were empirically set to strongly reward a positive rotation along the $z$-axis while penalizing other motion.

3 RESULTS

Figure 2 represents the average reward over time during the training process using the DRL framework. The network converged to a maximal value after approximately 11 million training steps. After training, the attached video demonstrates the ability to use DRL to map pixels to actions corresponding to rotation of the ball to perform in-hand manipulation.

4 CONCLUSION

In this paper, we presented an approach to perform in-hand manipulation without knowledge of the robot model or position. While our results are preliminary, they show promise in the ability
to perform highly dexterous tasks using DRL approaches. Our goal is to expand on this work by comparing results from a purely vision based approach to an approach that does include model and state information. In addition, while our current system does not leverage tactile sensors, we would like to compare our results when adding raw tactile sensor data as part of the state vector. Finally, our plan is to validate the results on a physical MPL and evaluate the amount of transfer learning that can be applied from simulation.

References


