Learning Dexterous In-Hand Manipulation
OpenAI (Andrychowicz, Marcin, et al.)

Markus Novak - Seminar Computational Intelligence A
January 14th, 2020
Overview

- Introduction
- Task and System Overview
- Transferable Simulations
  - Domain randomization
- Learning Control Policies From State
- State Estimation from Vision
- Results
Introduction

- Dexterous in-hand manipulation:
  - Shadow Dexterous Hand: 24 degrees of freedom
  - Modeling hardly possible, physical trials slow and costly
Task and System Overview

- Objective: Reorientation of object to desired target configuration in-hand
- System overview:
Transferable Simulations

- Simulation: coarse approximation of the real world
- Create a distribution over many simulations with randomized environment aspects: *domain randomization*

- Kinds of randomizations:
  - Observation noise
    - added Gaussian noise to policy observations
  - Physics randomizations
    - e.g. friction and damping coefficients, link masses, …
  - Unmodeled effects
    - model of motor backlash by delays and action noise
    - unmodeled dynamics: small random forces
Transferable Simulations

- Kinds of randomizations (cont.):
  - Visual appearance randomizations
Learning Control Policies From State

- Policy architecture
  - Training with Proximal Policy Optimization (PPO)
  - Short recap:  
    \[
    L^{CLIP}(\theta) = \mathbb{E}_t \left[ \min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right]
    \]
    - \( \hat{A}_t \approx A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t) \) … advantage estimate
    - \( r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \) … ratio of prob. of taking given action under current policy vs. old policy

---

**Algorithm 1** PPO, Actor-Critic Style

```plaintext
for iteration=1, 2, ..., do
  for actor=1, 2, ..., N do
    Run policy \( \pi_{\theta_{old}} \) in environment for \( T \) timesteps
    Compute advantage estimates \( \hat{A}_1, ..., \hat{A}_T \)
  end for
  Optimize surrogate \( L \) wrt \( \theta \), with \( K \) epochs and minibatch size \( M \leq NT \)
  \( \theta_{old} \leftarrow \theta \)
end for
```
Learning Control Policies From State

- Policy architecture (cont.)
  - Consists of 2 networks:
    - Value network, predicts discounted sum of future rewards from a given state
    - Policy network, maps observations to actions
    - Same architecture, different weights
  - All observations given to value network, only part to policy network
  - Use of recurrent neural network with memory (LSTM)
Learning Control Policies From State

- **Actions and Rewards**
  - Actions: Desired joints angles relative to the current ones
    - Discretized into 11 bins
  - Reward given at timestep $t$: $r_t = d_t - d_{t+1}$
    - $d_t$ ... rotation angles between desired and current object orientations
    - +5 reward whenever a goal is achieved
    - -20 reward whenever the object is dropped

- **Distributed Training**
  - Same PPO implementation as for OpenAI Five
  - 384 worker machines, each with 16 CPU cores, to generate experience $\rightarrow$ 2 years of simulated experience per hour
  - Single optimizer machine with 8 GPUs
State Estimation from Vision

- Policy needs object’s position as input → Vision model
- Trained only on synthetic data coming from simulator

- Model architecture:
  - Convolutional neural network
  - Input: 3 RGB cameras from different viewpoints

- Training:
  - 2 GPUs for rendering
  - 1 GPU for running the network and training
Results

- Summary of approach:

A. Distributed workers collect experience on randomized environments at large scale.

B. We train a control policy using reinforcement learning. It chooses the next action based on fingertip positions and the object pose.

C. We train a convolutional neural network to predict the object pose given three simulated camera images.
Results

- Summary of approach (cont.):
Results

- Qualitative results (emerged behavior):

Figure 7: Different grasp types learned by our policy. From top left to bottom right: Tip Pinch grasp, Palmar Pinch grasp, Tripod grasp, Quadpod grasp, 5-Finger Precision grasp, and a Power grasp. Classified according to [18].

- Policies naturally discover many grasp types and manipulation strategies found in humans
Results

- Quantitative results:
  - Measure: number of *consecutive* successful rotations until:
    - object is dropped
    - a goal has not been achieved within 80 seconds
    - until 50 rotations are achieved

- Results:

<table>
<thead>
<tr>
<th>Simulated task</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block (state)</td>
<td>43.4 ± 13.8</td>
<td>50</td>
</tr>
<tr>
<td>Block (state, locked wrist)</td>
<td>44.2 ± 13.4</td>
<td>50</td>
</tr>
<tr>
<td>Block (vision)</td>
<td>30.0 ± 10.3</td>
<td>33</td>
</tr>
<tr>
<td>Octagonal prism (state)</td>
<td>29.0 ± 19.7</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Physical task</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block (state)</td>
<td>18.8 ± 17.1</td>
<td>13</td>
</tr>
<tr>
<td>Block (state, locked wrist)</td>
<td>26.4 ± 13.4</td>
<td>28.5</td>
</tr>
<tr>
<td>Block (vision)</td>
<td>15.2 ± 14.3</td>
<td>11.5</td>
</tr>
<tr>
<td>Octagonal prism (state)</td>
<td>7.8 ± 7.8</td>
<td>5</td>
</tr>
</tbody>
</table>
Results

- Randomization effects:
  - All Randomizations
  - No Randomizations
  - No Observation Noise
  - No Unmodeled Effects
  - No Physics Randomizations

- On real robot:
Results

- Effect of memory:
  - LSTM Policy / LSTM Value
  - FF Policy / FF Value
  - FF Policy / LSTM Value

- On real robot:

<table>
<thead>
<tr>
<th>Network architecture</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM policy / LSTM value (state)</td>
<td>18.8 ± 17.1</td>
<td>13</td>
</tr>
<tr>
<td>FF policy / LSTM value (state)</td>
<td>4.7 ± 4.1</td>
<td>3.5</td>
</tr>
<tr>
<td>FF policy / FF value (state)</td>
<td>4.6 ± 4.3</td>
<td>3</td>
</tr>
</tbody>
</table>
Thank you for your attention!

Questions?

Video: https://www.youtube.com/watch?v=jwSbzNHGflM
Follow-up paper: https://openai.com/blog/solving-rubiks-cube/