Robot Assistance

Shelvin Pauly
University of Maryland, College Park

ABSTRACT
Autonomous robots have the potential to serve as versatile caregivers that improve the quality of life for millions of people worldwide. Yet, this is an untapped industry as it poses numerous challenges related to human robot interaction and other safety concerns. In this project, I have tried to simulate robot assistance for general caregiving purposes such as dressing and feeding.

Index Terms:
Proximal Policy Optimization (PPO), Soft Actor Critic (SAC), Activities of Daily Living (ADL), Assistive Gym, Dressing Assistance, Feeding Assistance, Training, Testing, Reward, Policy

1. INTRODUCTION:

Eating and dressing are the two most essential Activities of Daily Living (ADLs) for staying healthy and living at home independently. The project simulates the ADL of eating using the Sawyer robot and dressing using the Jaco robot. The initial motivation for this project was the highly grossing topic of robot assisted dressing. I started off working on the same but faced various issues including but not limited to hardware constraints. I finally ended up simulating just robot assisted feeding using different types of reinforcement learning algorithms and comparing the advantages and limitations of the same to analyze the scope for future work.

2. BACKGROUND:

One of the motivations for this project is aiding the people with special needs and the elderly.

2.1 LITERARTURE REVIEW:

Based on the limited research on previous works for feeding assistance and dressing assistance, [1] presents Assistive Gym, an open-source physics simulation framework for assistive robots that models multiple tasks. [2] presents a new meal-assistance system and evaluations of this system with people with motor impairments. [3] investigates the application of deep reinforcement learning techniques to robot-assisted dressing.

2.2 OBJECTIVE, ENVIRONEMNT, CONTROL:

Objective – To feed the human with food, i.e., to bring the spheres in the spoon close to the human model's head without spilling it or causing any kinds of discomfort for the human model.

The environment consists of a human on a wheelchair and a robot manipulator mounted on the wheelchair. The dressing assistance environment uses a hospital gown to dress the human with and, the feeding environment uses a bowl on a table and a spoon with some spheres that can be considered as the food. To automate these tasks of feeding and dressing, we need to be able to train the model for the robot control real time using reinforcement learning algorithms such as PPO, SAC, etc. and then use this trained model and feed it with inputs to be able to achieve the final goal/task.

3. IMPLEMENTATION:

Firstly, the environment is setup in Assistive Gym.

Assistive Gym – The Assistive gym is a high-level simulation framework that consists of interfaces to build and customize simulation environments with robots that can physically interact and assist people. An Assistive Gym environment is usually built using an open-source PyBullet physics engine. The PyBullet framework provides several benefits while simulating all kinds of physical human-robot interactions. It can also be used for real time simulations on both GPUs and CPUs, as well as soft bodies like cloth simulation. This is accomplished due to its ability to create robots and human models of various shapes, sizes, weights, and joint limits through programs. Assistive Gym directly integrates into the OpenAI Gm interface, allowing the usage of control policy algorithms like deep reinforcement learning that are already existent.

Create a vectorized and normalized environment (Multithreaded processing can be used, if available)

Train the policy and then save it along with the environment.

We can now use these saved environment and model for evaluation.

We also need a camera to render the outputs.

Finally, the evaluation loop can be started which keeps generating actions based on the policy. For each of these actions calculate the reward. Save each frame and after the loop ends, white it to an animated png.

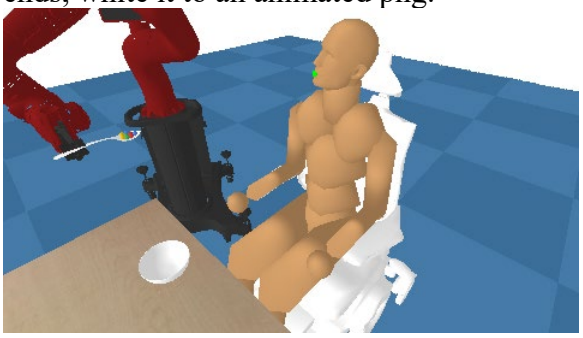

Figure 1: Basic Feeding Environment

Finally, the mean reward can be calculated.

3.1 PROXIMAL POLICY OPTIMIZATION (PPO) FOR FEEDING:

The Proximal Policy Optimization is essentially a modification of Trust Region Policy Optimization that learns from online data. This algorithm tries to simplify the optimization process while still keeping the advantages of the TRPO. PPO aims for a balance between ease of implementation, ease of tuning, sample complexity, sample efficiency and trying to compute an update at each step that minimizes the cost function while ensuring the deviation from the previous policy is relatively small.

$$
\begin{aligned}
&\textit{Algorithm for PPO using Actor} - \textit{Critic implementation}: \\
&\textit{Input}: \textit{ initial policy paramters} \\
&\textit{for iteration} = 1, 2, ...\textit{do} \\
&\textit{for actor} = 1, 2, ...\textit{do} \\
&\textit{Run policy } \pi_{\theta_{old}} \textit{ in environment for } T \textit{ timesteps} \\
&\textit{Compute Advantage estimates } \bar{A}_1, ..., \bar{A}_t \\
&\textit{end for} \\
&\textit{Optimize surrogate } L \textit{ wrt } \theta, \textit{ with } K \textit{ epochs and minibatch size } M <= NT \\
&\theta_{old} \leftarrow \theta \\
&\textit{end for}
\end{aligned}
$$

Figure 2. Pseudocode for PPO

3.1.1 TRAINING:

I trained the model for a total of 100,000-time steps. Each simulation rollout consists of 2048-time steps, where a policy can execute a new action at each time step. I performed a 10-epoch update of the policy after the actor completes a single rollout. Learning rate was set to be 0.0003, batch size – 64.

3.1.2 TESTING:

The agent is then evaluated for 100 evaluation episodes and gets the average reward. For testing we do not use the vectorized environment.

For PPO, I got-

Mean Reward: -3.70

Standard Reward: 69.89

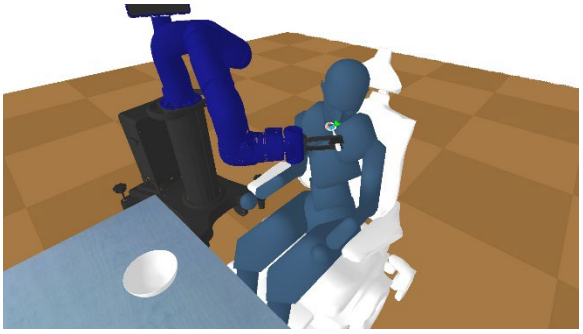

Figure 3. Testing the agent

3.2 SOFT ACTOR CRITIC (SAC) FOR FEEDING:

SAC is an off-policy algorithm. It is essentially the successor of Soft Q-Learning algorithm and incorporates the double Q-learning trick from Twin delayed deep deterministic policy gradient. It forms a bridge between stochastic policy optimization and DDPG-style approaches. A key feature is that it is trained o maximize a trade-off between expected return and entropy, a measure of randomness in the policy.

Algorithm 1 Soft Actor-Critic

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
repeat
  Observe state s and select action $a \sim \pi_\theta(\cdot|s)$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  If s' is terminal, reset environment state.
  if it's time to update **then**
    for j in range(however many updates) **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute targets for the Q functions:

$$y(r, s', d) = r + \gamma(1 - d)\left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s')$$

      Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s, a) - y(r, s', d))^2 \qquad \text{for } i = 1, 2$$

      Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta\left(\tilde{a}_\theta(s)|\, s\right)\right),$$

      where $\tilde{a}_\theta(s)$ is a sample from $\pi_\theta(\cdot|s)$ which is differentiable wrt θ via the reparametrization trick.
      Update target networks with

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho)\phi_i \qquad \text{for } i = 1, 2$$

    end for
  end if
until convergence

Figure 4. Pseudocode for SAC

3.2.1 TRAINING

I used all the same parameters for this algorithm as well and rest was kept to all the default values provided by stablebaselines.

3.1.2 TESTING:

The testing parameters are also kept the same for SAC algorithm

For SAC, I got-

Mean Reward: 11.32

Standard Reward: 80.16

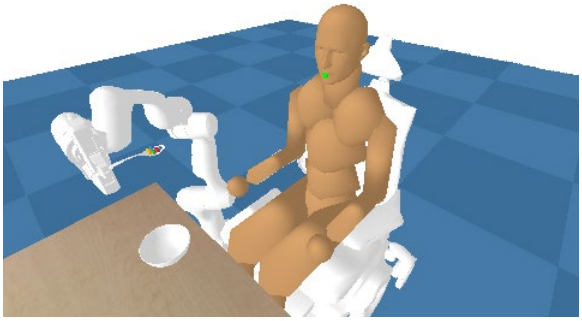

Figure 5. Testing the agent

3.3 ROBOT-ASSISTED DRESSING:

The goal of the robot is to pull the gown sleeve up a person's arm. A reward each is awarded for pulling up the sleeve, pulling the opening of the sleeve towards a person's hand and also for pulling the sleeve along the person's arm correctly towards the shoulder.

The most difficult part was the simulation of the dynamic cloth as it involved a lot of things like complex physics, cooperation, and risk of injury. There were a lot of problems visualizing the cloth during simulation. But the biggest setback proved to be the gap between the infinite training time it seemed to take, and the limited runtime google Collab provided, as the former was higher than the later. There seemed good probability that it could take up to 6 days for training because of the

dynamic cloth but still provide exceptionally low efficiency of around 26%. A lot of permutations in the way of training proved this efficiency could be greatly improved by changing the robot model itself.

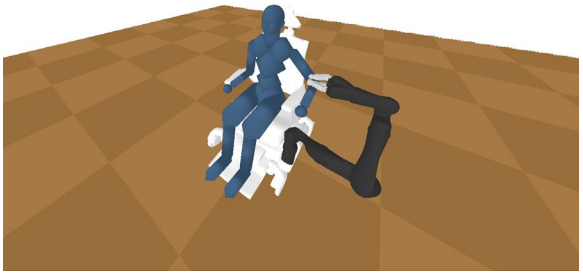

Figure 6. Dressing Environment

4. COMPARISON BETWEEN PPO vs SAC:

Both the algorithms, SAC and PPO, learn a policy and value function, but their strategies differ.

Firstly, PPO uses on-policy learning and SAC uses off-policy learning. On-policy algorithms tend to be more stable but data inefficient. Off-policy algorithms tend to be more unstable but data efficient.

Secondly, PPO augments exploration with entropy regularization. SAC algorithm strikes an exceptional balance between exploration and exploitation.

Finally, PPOs entropy regularization, encourages wider exploration and avoids convergence to a bad local minima. SAC's entropy maximization has the same advantage but also tend to give up on policies that choose unpromising behavior, which makes SAC to be more data efficient that PPO.

5. SIMULATION:

Actions are represented by changes in joint position.

The strength of the robot actuators is reduced to reduce the likelihood of applying high forces on the person.

At each time step, the robot records observations from the state of the system, executes an action according to a control policy, and then receives a reward.

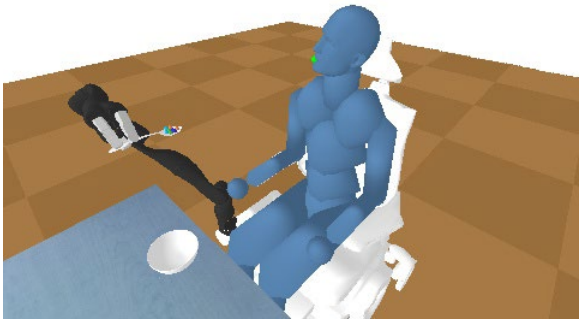

Figure 7. Trained agent

6. CONCLUSION:

Robot Assistance has been a field of massive interest in the recent times as it can highly augment the standard of living for many around us. However, there is still a lot of work to be done on these systems before it can be deployed in an environment to interact with humans independently. This project focuses on this very fact that there is massive potential in this concept, while also throwing light on factors like co-optimization with humans and other points of failure, as it can be hazardous to human life when implemented without proper research. During this project, I constantly wondered if it will ever be possible at all, as automated dressing seemed too complex. Even implementing the feeding task was hard as balancing the food on the spoon seemed a little tricky. I believe there are better modifications to the policies I have used. Hence, future work holds researching on those policies and producing a more robust model.

7. REFERENCES:

1. Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, C. Karen Liu, and Charles C. Kemp, "Assistive Gym: A Physics Simulation Framework for Assistive Robotics". https://arxiv.org/pdf/1910.04700.pdf

2. Daehyun Park, Yuuna Hoshi, Harshal P. Mahajan, Zackory Erickson, Wendy A. Rogers, Charles C. Kemp, "Active Robot-Assisted Feeding with a General-Purpose Mobile Manipulator: Design, Evaluation, and Lessons Learned". https://arxiv.org/pdf/1904.03568.pdf

3. Alexander Clegg, Zackory Erickson, Patrick Grady, Greg Turk, Charles C. Kemp, and C, Karen Liu, "Learning to collaborate from simulation for robot assisted dressing". https://arxiv.org/pdf/1909.06682.pdf

4. Lobbezoo, A.; Qian, Y.; Kwon, H.-J. Reinforcement Learning for Pick and Place Operations in Robotics: A Survey. *Robotics* 2021, *10*, 105. https://doi.org/10.3390/robotics10030105

5. I-DRESS, Assistive interactive robotic system for support in dressing, https://www.chistera.eu/sites/www.chistera.eu/files/CHIST-ERA%20Call%202014%20-%20RTCPS%20Topic%20-%20I-DRESS%202018.pdf

6. Daehyung Pak, Yuuna Hoshi, and Charles C. Kemp, "A multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-based Variational Autoencoder". https://arxiv.org/pdf/1711.00614v1.pdf

7. Mnih, V., Kavukcuoglu, K., Silver, D. *et al.* Human-level control through deep reinforcement learning. *Nature* 518, 529–533 (2015). https://doi.org/10.1038/nature14236.

8. PPO brief, https://www.geeksforgeeks.org/a-brief-introduction-to-proximal-policy-optimization/

9. PPO explained, https://spinningup.openai.com/en/latest/algorithms/ppo.html

10. SAC explained, https://spinningup.openai.com/en/latest/algorithms/sac.html