CIS II: Background Reading Report

Group 20: A reinforcement learning approach to robotic suturing

Team Members: Walee Attia (wattia1), Jocelyn Hsu (jhsu37), Jihoon Kim (jkim620)

Mentors: Dr. Anqi Liu, Dr. Adnan Munawar, Dr. Manish Sahu, Dr. Peter Kazanzides

March 14, 2023

Introduction

The use of surgical robotics has become increasingly popular in recent years. Robotic surgery offers several benefits over traditional open surgery, including improved precision, reduced blood loss, smaller incisions, and faster recovery times. However, robotic systems still require a human operator to control the instruments and make decisions during the procedure. Reinforcement learning (RL) is a type of machine learning that has the potential to improve robotic surgery by allowing robots to learn from their environment and make decisions autonomously.

Our project encompasses development of an RL platform specifically for automating robotic suturing with the da Vinci Research Kit (dVRK). In order to better understand the field of RL applied to the robotic surgery, we decided to explore applications of current RL algorithms in medicine as well as different RL paradigms.

In this report, we explore the use of RL in surgical robotics, with a particular focus on two paradigms: Deep Deterministic Policy Gradient (DDPG) [1] and Hindsight Experience Replay (HER) [2]. We then review recent research on the application of these paradigms in surgical robotics and their potential benefits and challenges as we synthesize the similarities between their research and our project. The primary papers that we will review are:

- M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," Advances in neural information processing systems, vol. 30, 2017. [2]
- V. M. Varier, D. K. Rajamani, F. Tavakkolmoghaddam, A. Munawar, and G. S. Fischer, "AMBF-RL: A real-time simulation based reinforcement learning toolkit for medical robotics," in 2022 International Symposium on Medical Robotics (ISMR). IEEE, 2022, pp. 1–8. [3]
- F. Richter, R. K. Orosco, and M. C. Yip, "Open-sourced reinforcement learning environments for surgical robotics," arXiv preprint arXiv:1903.02090, 2019. [4]

Reinforcement Learning [5]

Reinforcement learning involves an agent interacting with an environment to learn how to take actions that maximize a cumulative reward signal. The agent learns through trial and error, gradually improving its policy for taking actions based on the feedback it receives from the environment in the form of rewards or penalties.

The RL framework consists of an agent, an environment, a set of states S, a set of actions A, a reward function R, transition probabilities P, and a discount factor $\gamma \in [0, 1]$. The agent is responsible for taking actions in the environment based on its current policy, which maps states to actions. The environment provides feedback to the agent in the form of rewards or penalties, which the agent uses to update its policy. The state of the environment is typically represented as a vector of features that captures relevant information about the current situation.

The agent's goal in RL is to learn a policy that maximizes the cumulative reward over time. The cumulative reward for a step at time t can be computed with

$$R_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{1}$$

This is typically achieved through the use of a value function or a Q-function, which estimates the expected cumulative reward of following a given policy from a given state or state-action pair, respectively. The optimal value function $V^{\pi*}$ can be updated using the Bellman equation, where

$$V^{\pi*} = \max_a \{R(s,a) + \gamma\{P(s'|s,a\}V^{\pi*}(s')\} \tag{2}$$

which expresses the expected reward in terms of the current maximized reward and the expected future reward given the future state s'.

One common approach to RL is to use a deterministic policy, which maps each state directly to a specific action. This is in contrast to stochastic policies, which produce a probability distribution over actions for each state. Deterministic policies can be learned using methods such as the deep deterministic policy gradient (DDPG) algorithm, which uses deep neural networks to approximate the policy and the value function.

Deep Deterministic Policy Gradient [1]

The DDPG algorithm is a model-free, off-policy actor-critic algorithm that is specifically designed for continuous control problems, making it well-suited for robotic control tasks. The algorithm is an extension of the popular Q-learning algorithm used in RL that operates in continuous action spaces by learning a deterministic policy function that maps states to actions. The key idea behind DDPG is to use deep neural networks to approximate the value function and the policy function, allowing the algorithm to handle high-dimensional state and action spaces.

In DDPG, the actor network is responsible for selecting actions based on the current state, while the critic network evaluates the chosen actions and the resulting state transition. The actor network is trained using a deterministic policy gradient, which is the gradient of the action-value function with respect to the policy parameters. Parameters can be optimized by minimizing the loss

$$L(\theta^Q) = \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta, r_t \sim E}[(Q(s_t, a_t|\theta^Q) - y_t)^2] \tag{3}$$

where

$$\begin{aligned} &y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta^Q) \\ &\beta \text{ is the stochastic policy} \\ &\rho^\beta \text{ is the discounted policy} \end{aligned} \tag{4}$$

The critic network is trained using temporal difference (TD) learning, which involves updating the estimate of the action-value function based on the difference between the observed reward and the predicted reward.

One of the key benefits of DDPG is that it can learn policies that are deterministic, meaning that given a state, the policy will always produce the same action. This is in contrast to many RL algorithms that learn stochastic policies, which produce a distribution over actions. This determinism is particularly important for robotics applications, where precise control is often required. Additionally, DDPG can learn from experience stored in a replay buffer, which allows the algorithm to learn from past experiences and reduce the variance of the updates to the policy and value functions.

Overall, DDPG has proven to be a powerful algorithm for continuous control problems and has been successfully applied to a variety of robotic tasks, including manipulation, locomotion, and UAV control. However, like any RL algorithm, DDPG has major limitations: the need for careful hyperparameter tuning, sensitivity to the choice of the replay buffer size, and the potential for overfitting to the training data.

Hindsight Experience Replay [2]

HER is an extension of the DDPG algorithm that was specifically designed to address the challenge of sparse rewards in RL, which is a common problem in many robotics applications. The key idea behind HER is to modify the reward function during training so that even unsuccessful episodes can contribute useful information to the learning process. This is achieved by relabeling the original goal with the achieved goal in hindsight and using this modified goal to compute the reward.

In HER, a modified replay buffer is used to store not only the original transitions experienced by the agent, but also the modified transitions that result from relabeling the goal. During training, the agent samples transitions from this buffer and uses them to update the policy and value functions. By relabeling the goals in hindsight, HER is able to turn unsuccessful episodes into successful ones, providing the agent with more opportunities to learn and reducing the sparsity of the reward signal.

Algorithmic Approach

HER can be applied to RL agents learning a set of goals G. For all $g \in G$, there exists a state s such that the agent's goal g is achieved, or $f_g(s) = 1$. For some batch of iterations B, the states and their associated actions that achieved transitions between states can be stored in a replay buffer. After updating the replay buffer R, R is sampled such that the policy can be optimized based off the history of states and actions, regardless of the rewards of those actions.

Experiment & Results

One application of HER involves a bit-flipping environment with a large action space A such that $|A| = n > 40$. For any ith iteration, the ith bit will be flipped. In such an example, the authors note that traditional RL approaches cannot train such an environment because all actions will receive a reward of -1 and cannot efficiently conduct thorough exploration to achieve an efficient sequence of actions. With the addition of HER, the trajectory of the current state and sequence of actions can be examined with a modified goal, such that the agent learns how to achieve the current state. This additional learning achieves a much higher success rate (Fig. 1).

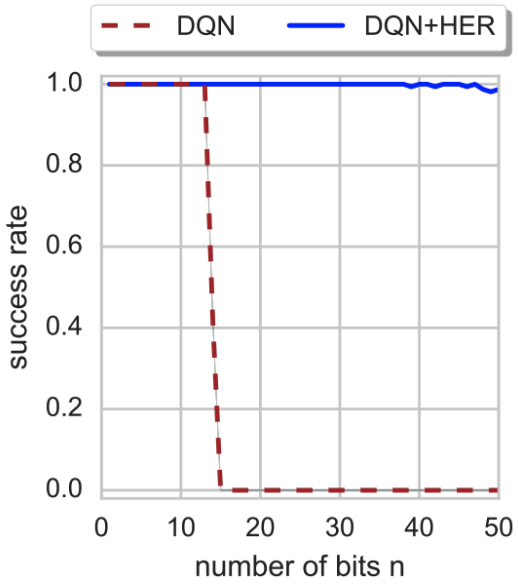

Figure 1: Success Rate Increase with Addition of HER [2]

RL Applications in Medicine

Here we outline three papers presenting findings crucial to our robotic suturing project. *A Real-Time Dynamic Simulator and an Associated Front-End Representation Format for Simulating Complex Robots and Environments* covers a simulator compatible with the da Vinci Research Kit (dVRK) used to model medical robotics. *Open-Sourced Reinforcement Learning Environments for Surgical Robotics* develops a reinforcement learning toolkit, dVRL, and demonstrates successful agents trained with dVRL.

A Real-Time Dynamic Simulator and an Associated Front-End Representation Format for Simulating Complex Robots and Environments [3]

The ability of reinforcement learning techniques to solve problems within robotics relies on the presence of robust simulation frameworks. The Asynchronous Multibody Framework (AMBF) is a real-time dynamics simulator that allows for such progress in the medical robotics domain. In this paper, Vignesh Manoj Varier, Dhruv Kool Rajamani, Farid Tavakkolmoghaddam, Adnan Munawar, and Gregory S Fischer present AMBF-RL (ARL), an RL toolkit that enables the design of control algorithms and the collection and processing of expert data from demonstration on real systems. Moreover, they validated the toolkit by simulating a debris removal using a reach task on the da Vinci Research Kit (dVRK) Patient Side Manipulator (PSM) arm. This was done using by finding the optimal RL policy using both DDPG and DDPG + HER (Deep Deterministic Policy Gradient and Hindsight Experience Replay) models before successfully transferring it to the physical system.

Methods

ARL integrates AMBF with the OpenAI Gym by using its interface. The toolkit allows for DDPG, DDPG + HER, and GAIL (Generative Adversarial Imitation Learning) model development. The environment is suited for training and simulating on the dVRK PSM arm but can be extended to other robots. The API that allows for the above utility consists of three main functions:

- Reset: Randomly chooses a new desired goal position and returns the end-effector arm Cartesian coordinates to base.
- Init: Initializes AMBF python client which is used to get a handle of the robot. The state space and action space are also initialized. Many parameters can be specified via the input to this method including:
  - Goal threshold (margin which L2 norm distance must be smaller than to activate reward)
  - Limits for the action space (how far in the x, y, and z directions in which you can step)
  - Joint specifications for which ones the client should control
  - Joint limits (specify the extent to which each joint can move)
- Step: Given the state of the environment, select and execute an action. States are updated by computing the changes in robot kinematics following the action.

After training and configuring the environment, the model can be saved for specific iterations, allowing thorough evaluation of the trained environment (Fig. 2)

Experiments & Results

The ARL toolkit was validated by performing a reach task portraying debris removal on the simulated dVRK PSM before transferring the learned policy to the physical ARM. The observation space consisted of the Cartesian coordinates of the end-effector (position) and the arm's velocity. A desired goal position was chosen at random with the constraint that it was in the defined workspace limits of the PSM arm. The reward function revolved around the L2 norm of the difference between the desired goal (the aforementioned randomly chosen position) and achieved goal (where the end-effector is at a given step). An episode was determined to be successfully completed when this distance is less than 10mm.

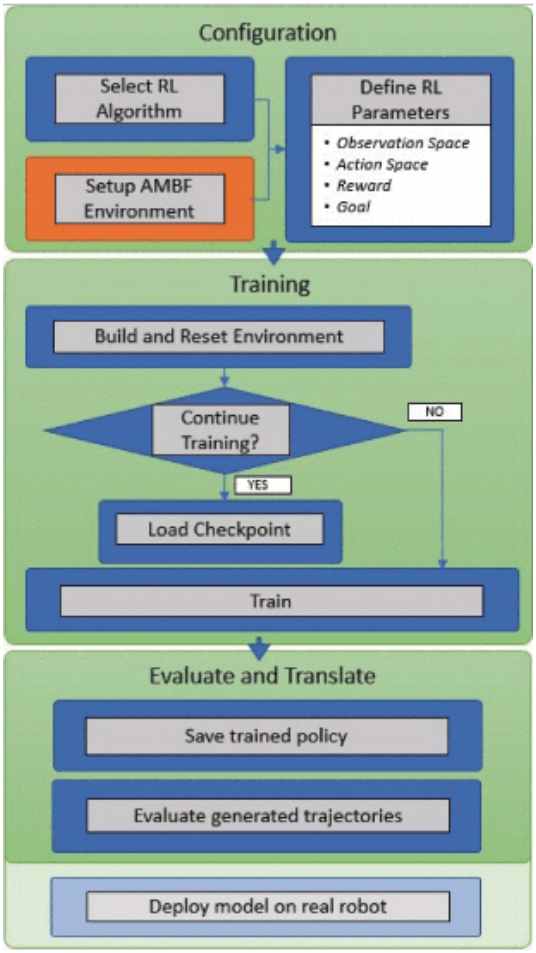

Figure 2: AMBF RL Model Development Pipeline [3]

Two different reward schemes and methods for finding the optimal policy were used. Under the sparse reward scheme, a positive reward was returned if the L2 norm distance at the given step was less than a set threshold, and a nonpositive (0 or -1 depending on the type of model) reward was returned otherwise. Under the continuous reward scheme, the reward is as follows:

$$Reward = 1 - \frac{d * .5}{d_{max}} \tag{5}$$

where d refers to the L2 norm distance and d_max refers to the maximum distance within the workspace. The two models that were used DDPG and DDPG + HER.

Different trials were run where the maximum number of steps per episode was varied. The corresponding success rates were observed, defined by the percentage that the PSM arm reached the desired goal (within 10 millimeters) over the course of 20 trials (Fig. 3). The authors also show how the reward changes over time as the agents learns across increased number of steps (Fig. 4).

Overall, the DDPG + HER model outperformed the DDPG model. Fig. 3 demonstrates that the success rate of DDPG + HER is consistently above DDPG regardless of the set maximum number of steps per episode. Fig. 4 shows that the reward quickly increases for DDPG + HER before flattening out whereas the reward for DDPG fluctuates significantly even at the higher number of steps, indicating unsuccessful learning.

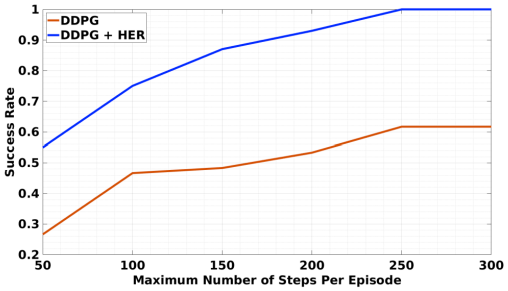

Figure 3: Success rate vs Maximum num steps per episode — DDPG vs DDPG + HER [3]

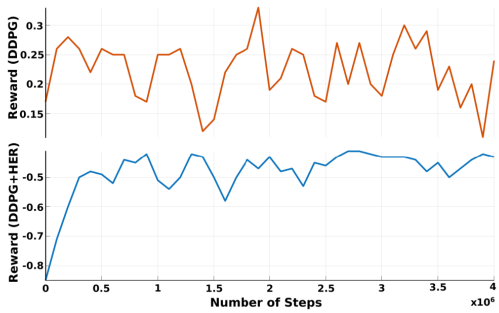

Figure 4: Reward vs Number of steps — DDPG vs DDPG + HER [2]

Open-Sourced Reinforcement Learning Environments for Surgical Robotics [4]

Reinforcement learning has become an increasingly popular method to solve complex domain-specific problems. In this paper, Richter, Orosco, and Yip introduce a reinforcement learning simulation environment compatible with dVRK, called dVRL. The goals of the environment are two-fold: to provide a reinforcement learning environment for training surgical robotics, and to transfer the learned policies from simulation to an actual robot.

Methods

In order to create the environment, the simulation relies on Markov Decision Processes to train the agent to reach its goal through a series of actions. For each action the agent takes, it will evaluate changes to the environment; the agent will proceed to select another action that maximizes the reward, and the cycle repeats until the agent reaches its end goal.

For the simuation, dVRK relies on OpenAI gym environment[6] and the V-REP physics simulator [7] for modeling the Patient Side Manipulator (PSM) arm. The PSM is defined by its end-effector position $\mathbf{p_t}$ in base frame and jaw-angle j_t and its bounded in its environment with center $\mathbf{p_c}$ such that

$$[\mathbf{p_c}]_i - \rho \leq \mathbf{p_t}]_i \leq [\mathbf{p_c}]_i + \rho \tag{6}$$

for each dimension i and $0 \leq j_t \leq 1$. After evaluating the best actions Δ_t and ϕ_t to take at time t, the PSM is updated such that

$$\begin{aligned} \mathbf{p_{t+1}} &= \rho\Delta_t + \mathbf{p_t} \\ j_{t+1} &= (\phi_t + 1)/2 \end{aligned} \tag{7}$$

The ρ term ensures effective transferring of policies of simulation to real environments to account for overshooting and instabilities.

The authors developed two overarching movements: fetch and pick. For fetch, to move the end effector to the goal position g, the authors developed state space $s_t = [\tilde{\mathbf{p_t}}\ \tilde{\mathbf{g}}]$ and action $a_t = [\Delta_t]$ with a simple reward function when

$$r(s_t) = \begin{cases} -1 & \text{if } \rho \parallel \tilde{\mathbf{p_t}} - \tilde{\mathbf{g_t}} \parallel > \delta \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

For picking an object at $\mathbf{o_t}$, $s_t = [\tilde{\mathbf{p_t}}\ 2j_t - 1\ \mathbf{o_t}\ \tilde{\mathbf{g}}]$ and $a_t = [\Delta_t\ \phi_t]$. The reward function defined similarly as

$$r(s_t) = \begin{cases} -1 & \text{if } \rho \parallel \tilde{\mathbf{o_t}} - \tilde{\mathbf{g_t}} \parallel > \delta \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Experiments & Results

The Reach and Pick environment were evaluated for accuracy and efficiency. Deep Deterministic Policy Gradients (DDPG) was employed to minimize Bellman Loss $L_Q = (Q(s_t, a_t) - (r_t + \gamma Q(s_{t+1}, a_{t+1}))^2$ by approximating the policy and reward function. The policy is then optimized with the minimization of $L_\pi = -\mathbb{E}_{s_t}[Q(s_t, \pi(s_t)]$. Hindsight Experience Replay (HER) was also incorporated for increasing efficiency of training and updating goals based on previously achieved states.

The environments were then tested on real-world simulations, using the Pick environment to navigate the PSM to a sponge and grasp it in its end-effector and the Reach environment to orient a suction and irrigation tool. A third task, suction and debris removal in an abdominal model, involved the combination of both the Pick and Reach environments. The pick was able to reach for objects within 100 steps with 100% success rate with the addition of HER; without HER and behavior cloning, the pick environment was not trainable (Fig. 5). The PSM Reach environment achieved approximately 85% accuracy given 100 steps and was capable of achieving 100% success rate given 1000 simulation steps. Both reach and pick were able to reach the end goal within 2-3.5 seconds per rollout of each environment (Fig. 6).

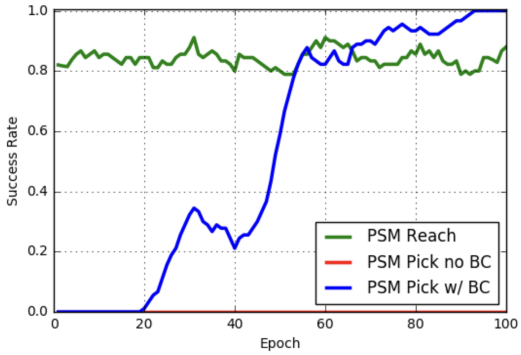

Figure 5: Success of dVRL Pick and Reach Environments [4]

TABLE II:
Timing Results of one rollout per Environment

Num. of Env.	PSM Reach	PSM Pick
1	2.09 sec	2.09 sec
2	2.36 sec	2.35 sec
4	2.78 sec	2.78 sec
6	3.03 sec	3.02 sec
8	3.27 sec	3.26 sec

Figure 6: Efficiency of dVRL Pick and Reach Environments [4]

Discussion

Analysis of Findings

One of the key benefits of HER is that it can be easily integrated with existing RL algorithms, including DDPG [2]. In fact, adding HER to DDPG can significantly improve the performance of the algorithm on a variety of robotic control tasks, including object manipulation and locomotion.

However, like any algorithm, HER has some limitations and challenges. One potential issue is the choice of the relabeling strategy, which can have a significant impact on the performance of the algorithm. Additionally, HER can be computationally expensive, as it requires storing and sampling from a large replay buffer. Finally, the effectiveness of HER may be limited in environments with highly structured or constrained goals, where relabeling the goal in hindsight may not be a meaningful operation [2].

Despite the limitations, HER is effective at training multi-goal environments, as shown in experiments conducted with AMBF-RL and dVRL toolkits. For example, the AMBF-RL paper highlighted the impact of combining HER with DDPG as the authors noted a significant increase in success rate for the reach / debris removal task with the DDPG + HER model relative to the DDPG model. Given the simplicity of the reach task and therefore the associated reward function (which revolves around the distance between the achieved and desired goal), it would be interesting to see the results of the ARL toolkit on more complicated tasks that require more sophisticated reward processes.

The dVRL environment is built upon computations with respect to the end-effector coordinate frame system; this simplifies the number of variables with the and it versatile for various tool attachments. The environments built achieved high accuracy, even in realistic surgical environments. However, the tasks chosen in the paper are relatively simple, and further training and evaluation is needed for more complex surgical tasks such as automated suturing. This will decrease efficiency of the dVRK system, increasing the number of steps needed and requiring more training epochs for sufficient learning.

Incorporation into our project

With our background readings, we can incorporate results of these studies into our project. To develop our RL algorithm, we will incorporate DDPG and HER when selecting the next step to perform in the suturing process that maximizes the reward. Not only is HER compatible with DDPG and built to augment RL algorithms specifically, but it has also been shown to increase learnability of models in both the AMBF and dVRL studies.

The OpenAI Gym environment that we are designing will be inspired by the setup of AMBF and dVRL, both of which have publicly available code to reference. Our goal is to construct an efficient and accurate autonomous suturing environment, and build the environment transferable such that it can transferable for other da Vinci Research Kit RL projects.

References

[1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: https://arxiv.org/abs/1509.02971

[2] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[3] A. Munawar, Y. Wang, R. Gondokaryono, and G. S. Fischer, "A real-time dynamic simulator and an associated front-end representation format for simulating complex robots and environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Nov. 2019. [Online]. Available: https://doi.org/10.1109/iros40897.2019.8968568

[4] F. Richter, R. K. Orosco, and M. C. Yip, "Open-sourced reinforcement learning environments for surgical robotics," *arXiv preprint arXiv:1903.02090*, 2019.

[5] D. Silver, "Introduction to reinforcement learning with david silver." [Online]. Available: https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver

[6] OpenAI, "Openai gym documentation: Basic usage." [Online]. Available: https://www.gymlibrary.dev/content/basic_usage/

[7] G. A. Fontanelli, M. Selvaggio, M. Ferro, F. Ficuciello, M. Vendittelli, and B. Siciliano, "A v-rep simulator for the da vinci research kit robotic platform," in *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, 2018, pp. 1056–1061.