AMBF-RL: A real-time simulation based Reinforcement Learning toolkit for Medical Robotics

Vignesh Manoj Varier[1,*], Dhruv Kool Rajamani[1,*], Farid Tavakkolmoghaddam[1], Adnan Munawar[2], Gregory S Fischer[1]

2022 International Symposium on Medical Robotics (ISMR) | 978-1-6654-6928-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISMR48347.2022.9807609

***Abstract*— Recently, Reinforcement Learning (RL) techniques have seen significant progress in the robotics domain. This can be attributed to robust simulation frameworks that offer realistic environments to train. However, there is a lack of platforms which offer environments that are conducive to medical robotic tasks. Having earlier designed the Asynchronous Multibody Framework (AMBF) - a real-time dynamics simulator well-suited for medical robotics tasks, we propose an open source AMBF-RL (ARL) toolkit to assist in designing control algorithms for these robots, as well as a module to collect and parse expert demonstration data. We validate ARL by attempting to partially automate the task of debris removal on the da Vinci Research Kit (dVRK) Patient Side Manipulator (PSM) in simulation by calculating the optimal policy using both Deep Deterministic Policy Gradient (DDPG) and Hindsight Experience Replay (HER) with DDPG. The trained policies are successfully transferred onto the physical dVRK PSM and tested. Finally, we draw a conclusion from the results and discuss our observations of the experiments conducted.**

I. Introduction

Reinforcement Learning (RL) techniques have seen significant progress in the robotics domain [1]. Open-source RL frameworks such as OpenAI: Gym [2] and Baselines [3] have enabled effortless implementation of complex algorithms in simulation and real robots. The availability of robust simulation platforms that offer realistic simulation environments and easy integration with these RL frameworks have contributed to the increased adoption of these techniques. Most of these platforms also offer the ability to transfer trained models to the real robots with minimal modifications [4]. This added benefit reduces the engineering effort and cost associated with the direct implementation of RL on physical robots [5].

While the utilization of learning-based techniques have proliferated in most domains of robotics, they remain scarce in the field of medical robotics. Recent efforts have proposed automating surgical tasks in Robot Assisted Surgeries (RAS) to circumvent frequent clutching [6], eliminate heuristic tuning and processing high-dimensional data [7], reducing the cognitive load on surgeons [8], [9] and utilizing simulations for collection of large amounts of data [10]. Despite these significant efforts, there remains a disconnect in making these techniques accessible to a broader range of researchers, as access to the robotic platforms is not feasible for a large portion of the medical robotics community.

Previously, we developed the Asynchronous Multi-Body Framework (AMBF) [11] - a real-time dynamics simulator geared towards medical robotic applications, capable of simulating soft tissue [12], and offering a suite of surgical robot models like the da Vinci Research Kit (dVRK) [13], Raven II [14], and Neuro Robot [15]. In this paper, we propose an open source RL toolkit AMBF-RL (ARL)[1] that integrates with AMBF. We present the design architecture and features that can significantly enhance the implementation of RL algorithms for medical robots. We develop a module to collect expert-demonstrated data from dVRK Patient Side Manipulator (PSM) to use for tasks that involve Learning from Demonstration [16], [17]. We validate the ARL toolkit by performing a reach task that portrays debris removal in an operating room. Deep Deterministic Policy Gradient (DDPG) [18], and Hindsight Experience Replay (HER) with DDPG [19] are implemented to obtain policies to complete this task. The trained policies are successfully transferred to the dVRK PSM to reach the desired goal.

II. Background

Researchers have made substantial efforts in applying RL based techniques to control robots, specifically in the fields of manipulation [20], [21] and mobility [5], [22]. There are some common problems that are associated with training on physical robots such as restricted environments [1], costly roll-outs [23], and an emphasis on safe exploration due to hardware constraints [24]. To circumvent these problems, realistic models of robots are created and trained in simulation. Once a satisfactory RL model is trained, the model can be transferred onto the real robot (*Sim2Real*) [4]. Additionally, training in a simulation environment has the benefit of faster training which can be leveraged by accelerating the simulated physics and/or launching multiple training instances to overcome the exploration-exploitation dilemma [1].

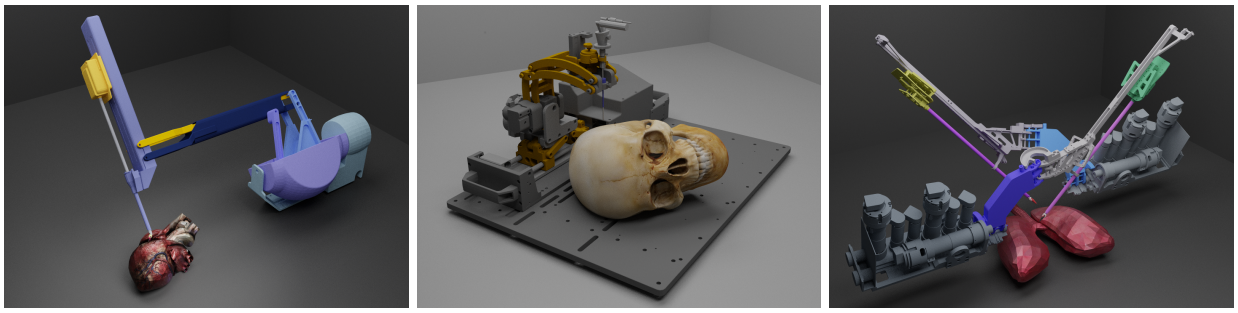
Although RL training environments are prevalent in robotics, they can be improved to increase their adoption in medical robotics. Recent efforts in automating surgical tasks using RL are limited to a few tasks such as cutting [25], suction [26], and suture hand-off [16] which demonstrate a growing demand for a framework that offers environments tailored

[1]Authors with the Robotics Engineering Department, Worcester Polytechnic Institute, WPI, 100 Institute Road, MA 01609, USA {vvarier, dkoolrajamani, ftavakkolmoghadd, gfischer}@wpi.edu

[2] Author with Laboratory for Computational Sensing and Robotics, Johns Hopkins University, Baltimore, Maryland 21218, USA {amunawar}@jhu.edu

* Authors had equal contributions in preparation of this manuscript.

[1]https://github.com/WPI-AIM/ambf_rl



(a) dVRK PSM (b) Neuro Robot (c) Raven II

Fig. 1: Examples of surgical robots modeled in AMBF.

towards medical robotics. Other works by Tan *et al.* [27] and Baek *et al.* [28] employ RL techniques for residency training and later automated resection using the V-REP [29] simulator. Researchers [30] have also demonstrated RL based surgical planning for cutting of soft tissue by deriving a tensioning policy to improve cutting accuracy. In 2019, Richter *et al.* [26] developed the first open-sourced RL environments for surgical robotics called dVRL and recently Xu *et al.* [31] designed a simulation platform for surgical robot learning that can be utilized for dVRK. To the best of our knowledge, dVRL and SurRoL are the only source RL environments for surgical robotics. Therefore, the following text provides an overview of dVRL and SurRoL and the motivation behind designing an RL toolkit for medical robotics using AMBF simulator.

A. Current RL environments available for Surgical Robots

dVRL offers open source environments that extend the OpenAI Gym framework by interfacing it with V-REP [29]. In their paper, the authors demonstrated debris removal and suction in simulation and transferred the learnt model onto a physical dVRK system to validate their environments. SurRoL was built based on PyBullet [32], which is a Python wrapper for Bullet physics. The authors utilized the dVRK meshes available from AMBF to train the dVRK PSM and ECM to perform tasks such as needle reach, needle pick, needle regrasp and tracking using ECM. Despite its novelty, the dVRL and SurROL has shortcomings that include (i) few environments for surgical tasks, (ii) complexity while incorporating soft-body dynamics due to the underlying simulation framework used in dVRL - VREP, and that (iii) only offers environments for the dVRK PSM and ECM. AMBF and ARL improve upon these shortcomings by providing more surgical environments, the capability to simulate surgical tasks on soft tissue, and a larger variety of medical robot models. To maintain consistency in validating the ARL toolkit, we performed the task of debris removal with a simulated dVRK PSM, and transferred the model onto the physical system.

B. Asynchronous Multi-Body Framework (AMBF)

AMBF is an open-source dynamic simulator that offers real-time simulation and haptic interaction of multi-bodies including robots, free bodies, and puzzles with support for training Neural Network for high-level control of robots [11]. The simulator utilizes and extends CHAI-3D [33] to provide real-time haptic interaction of multiple Input Interface Devices (IIDs). Examples of supported devices include the dVRK Master Tool Manipulator (MTM) and Razer Hydra Game Controllers in addition commercial haptic devices already incorporated by CHAI-3D. AMBF also offers support for simulation and interaction of soft bodies [12]. This capability is crucial in applications where manipulation and interaction with soft tissue are required (eg., suturing, cutting, pinching). AMBF provides a complementary Python client which offers a low barrier to entry interface with AMBF and a medium for training using Artificial Intelligence techniques.

AMBF simplifies describing a robot through its own front-end human-readable format (ADF) [11]. There are no limitations regarding the number of children or parents for a given joint in ADF in contrast to the more conventional spatial tree structure formats such as in URDF or SDF. This allows for a simplified description of parallel mechanisms which is advantageous for surgical robot simulation applications where most robots possess some form of closed-loop mechanisms.

Finally, several medical robots such as the dVRK (Fig. 1a), MRI-compatible Neurorobot [15] (Fig. 1b), and Raven II [14] (Fig. 1c) have already been modeled and provided as examples in AMBF. The ease of modeling medical robots in AMBF and the extensibility of the framework to provide realistic interactions between the robot and the environment provides an accessible platform for testing and development of new learning algorithms and control schemes on robots.

III. Architecture

This section provides an overview of AMBF's communication architecture and its incorporation with ARL. Next, the design overview of ARL is presented - it emphasizes on components of AMBF that enhance the functioning of ARL and provide an overview of its workflow. This section also expands on features specific to surgical robotics for the simulator (AMBF).

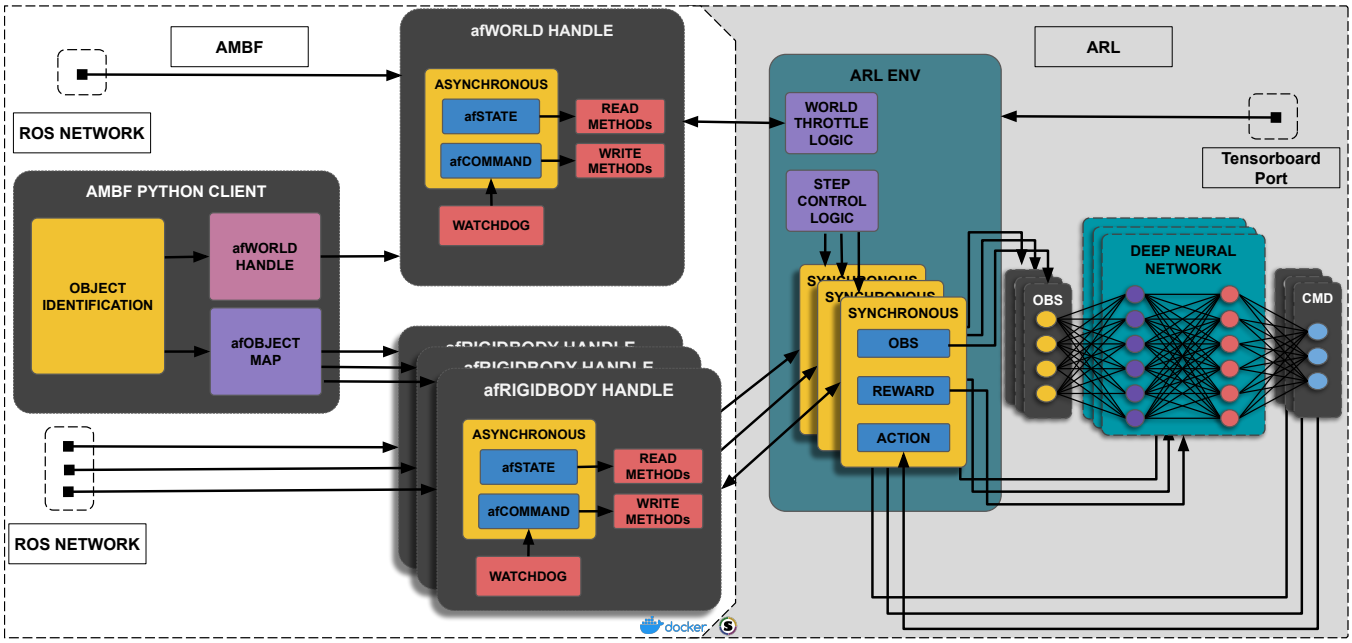


Fig. 2: Design of the AMBF Python Client. The identified **afObjects** are segregated from each other and any *afObject* can be queried by the **ARL ENV** class which wraps the AMBF Python Clients API and exposes an OpenAI Gym compatible API.

A. AMBF: Features that enhance ARL

AMBF utilizes Robot Operating Software (ROS) [34] as the communications middleware. In its implementation, all objects (bodies, sensors, cameras, lights etc.) (*afObjects*) own an exclusive instance of a base class called *afObjComm*. Similarly, the World object (*afWorld*) owns an instance of *afWorldComm*. Each communication instance is dispatched in a separate thread where the publishing (*afState*) and subscribing (*afCommand*) of ROS topics is performed. For the subscribers, rather than utilizing the default Spinner implementation, Custom Callback Queue are used. This provides encapsulation for the flow of data between different instances. For the *afState* and *afCommand* of each type of *afObject*, custom message payloads[2] are defined.

The distributed and asynchronous communication architecture of AMBF is leveraged by the AMBF Python client, shown in Fig. 2. The client identifies **afObjects** and dispatches exclusive threads for handling them. By doing so, the Python client encapsulates the data and the control of data-flow and thus provides an implementation where low over-head instances of various **afObjects** can be queried in isolation. This provides for an easy integration with RL frameworks and accomplished using the **ARL ENV** class (Fig. 2).

Since the underlying AMBF communication architecture is asynchronous, the **ARL ENV** class implements a step control mechanism to synchronize between actions, states and rewards. Additionally, the active simulation in AMBF can be dynamically throttled to compensate for the communication and training-related delays. This is achieved by setting a throttle flag, the number of jump steps and a clock in the *afCommand* payload of *afWorld* to step the simulation as shown in Fig. 3. This process is automated by the **ARL ENV**.

AMBF also supports speeding up the simulation to increase training speed and a headless mode (no Graphical User Interface). These modes are accessible via Command Line Interface (CLI) arguments at launch. The headless mode not only reduces the load on system resources (Fig. I), but also allows for the simulation to be run on window-less servers - a growing need due to the high resource usage required for training neural network models. rosbag's have been integrated with **ARL ENV** to allow for a standardized data storage and replay mechanism.

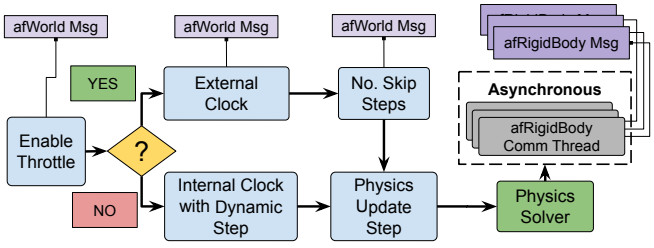


Fig. 3: Interface for dynamically throttling the simulation in AMBF using the **afWorld**'s *afCommand* message payload.

B. AMBF Reinforcement Learning (ARL) Environments

ARL integrates AMBF with OpenAI Gym by leveraging its API. ARL currently provides environments for DDPG,

[2] https://github.com/WPI-AIM/ambf/tree/ambf-1.0/ambf_ros_modules/ambf_msgs/

	Memory Usage (GB)	CPU Usage (%)		
	Average, Maximum, Minimum	*Average*	*Maximum*	*Minimum*
With GUI	0.7	18.08	19.25	16.67
Without GUI	0.5	9.83	8.83	10.91

TABLE I: CPU and Memory usage of ARL with and without Graphical User Interface (GUI). Both profiles were run on the same computer with the same background and foreground processes running. Specifications: AMD Ryzen 5 3600 6-Core Processor (12 threads) and 32 GB Corsair Vengeance DDR4 RAM

DDPG+HER and Generative Adversarial Imitation Learning (GAIL) [35] to train with the dVRK PSM Robot. This API is presented below and can be implemented to add more robots. Depending on the RL algorithm, either the main Gym environment (list of observation states) or goal Gym environment can be extended. For the goal Gym environment, the state of the robot consists of a dictionary containing: observation (current observations of the state), achieved goal (current result attained by the robot) and desired goal (final result to be achieved).

- **Reset**: This method resets the robot to home position (preset by user) and generates a goal position using random sampling
- **Init**: This method creates an instance of the AMBF Python client which is used to get a handle to the desired robot. Next the state and action space is initialized based on the input parameters. Users also need to define which joints of the robot need to be controlled by the Python client.
  Input: *action space limit*, *joints to control*, *goal position range*, *position error threshold*, *goal error margin*, *joint limits*, *workspace limits*, *enable step throttling*
- **Step**: This method gets the current state of the robot and takes the action. The next state is found by computing the forward and inverse kinematics of the robot. Then the robot is commanded to the desired state and the resulting reward is computed. While the next state is computed, users also need to ensure that the Kinematic and Dynamic constraints of the robot are also defined in this method.
  Input: *action*
  Returns: *Observation*, *Reward*, *Done*, *Info*

Stable-Baselines [36] provides an interface that could be used to define RL model parameters such as the actor learning rate, number of training steps and number of exploration steps. Fig. (4) shows the workflow followed to train a model. The make command creates an object handle to the spawned robots in AMBF through its Python client and throttles the simulation when enabled. For training, it is possible to either load a previously trained model and continue training or start training a new model. Finally, the model is saved and evaluated. Users can also provide specific values for roll-out, training and evaluation of their model. In the roll-out phase, the agent explores and populates the replay buffer with state transition parameters until the maximum number of roll-out steps. Then the model gets trained on the data which is randomly sampled from the replay buffer. Additionally, an evaluation environment can be used to test the learnt policies at regular intervals.

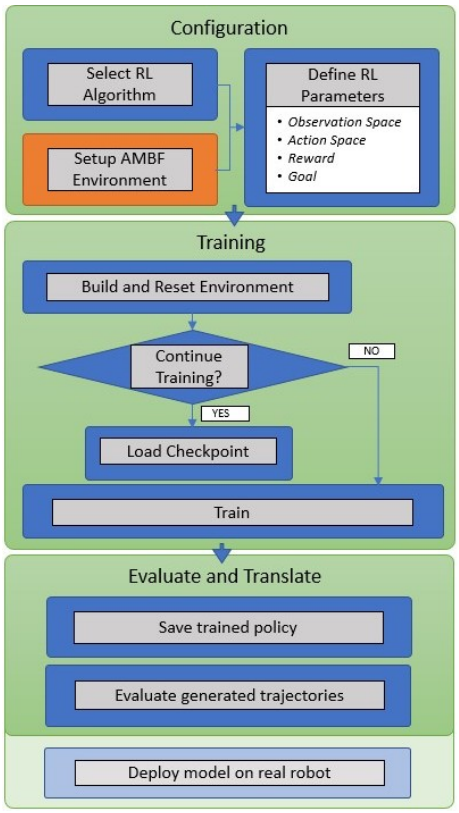


Fig. 4: Workflow for Creating, Training and Evaluating a RL Model.

C. Creating Expert Data for RL

Continuous space RL algorithms suffer from exploration inefficiency leading to increased training time or erroneous results [7], [17]. A possible workaround to tackle this issue would be using user recorded data to train the RL model initially. Therefore, a module was developed to bag ros data from AMBF and process the data in a format that could be used to train algorithms such as GAIL. The process for creating expert data is simple - while the expert is using the dVRK's MTM, the rostopics of the PSM's joint trajectories are bagged. After that a dictionary of Observations, Actions, Rewards, Episode returns and Episode Starts are generated by discretizing the collected data. A discretization factor can be set to change the accuracy when comparing the discretized data with the collected data. The collection of data using the provided module could potentially help improve the training efficiency and perform complex surgical tasks such as picking a suturing needle etc .

D. Containerised templates for high compute training

ARL also offers containerised environments that can be deployed on workstations or High Performance Clusters (HPC) using either docker or singularity. The container launches 4 processes independently - a roscore instance, a headless AMBF process, a training script which wraps the ARL Environment, and a tensorboard script to view the status of training in real-time. The user has the option to bind local ports to the remote server and/or the container host to subscribe to ROS topics being published in real-time. By loading the same robot model in AMBF, ARL allows playback of joint states as **afCommand** messages for **afObjects**. This allows local AMBF clients to render and jog robots that are undergoing training on a cluster. Similarly, another port can be bound to the server and/or container to view tensorboard logs in real-time.

IV. Experiment

A. Description

We validated ARL by performing a reach task that portrays debris removal surgical task with a simulated dVRK PSM in AMBF and transferred the learnt policy to the physical dVRK PSM. Two models were trained to obtain an optimal policy to perform this task - DDPG, and DDPG+HER. A simple observation space was selected for both RL models consisting of the end-effector's Cartesian position and velocity. For simplifying the training process, the workspace limits were set to $\mathbf{S} = \{s|x \in [-60mm, 55mm], y \in [-50mm, 60mm], z \in [-200mm, -90mm]\}$.

A desired goal position was randomly selected with the only constraint that it lies within the dVRK PSM's reachable workspace. Since RL is a goal based learning technique, the reward function was computed as the $L2$ norm of the distance between the tip of the end-effector (achieved goal) and the desired goal. We identified parameters that could significantly influence the reward and shaped the reward according to Alg. 1. An episode is successfully completed when the computed $L2$ norm between the achieved and desired goals is within $10mm$. We explored 2 separate reward schemes for this experiment - a sparse reward scheme, and a continuous reward scheme. In the case of the sparse reward scheme, a reward of $\{+1/0\}$ was given for success and failure respectively for DDPG; Similarly, a reward of $\{+1/-1\}$ was given for DDPG+HER. In contrast, the continuous reward scheme used a common reward for both models - the reward received by the agent directly depended on the $L2$ norm between current end-effector's tip position and the desired goal position. The action space size was chosen to be 3 (corresponding to the desired Cartesian position of the end-effector). To ensure stability of RL algorithms during training, the action space limits for RL algorithms were set to be symmetric and normalized between values $[-1, 1]$. These values were then scaled down to a limit of $[-5mm, 5mm]$ with a resolution of $1mm$. Therefore, the action space was set to $\mathbf{A} = \{a|x \in [-5mm, 5mm], y \in [-5mm, 5mm], z \in [-5mm, -5mm]\}$.

Since the learnt policy resulted in desired Cartesian positions, the values were directly applicable to the real dVRK using the CISST-SAW Libraries [37], [38].

Algorithm 1: Reward Function Definition for ARL dVRK PSM Environment

function `Reward`(*achievedgoal, desiredgoal, info*)**:**
- Compute distance between desired and achieved goal
- **if** *Sparse Reward*$_1$ **then**
  - **if** *Computed Distance <Goal Error Margin* **then**
    - *Reward = 1*
  - **else**
    - *Reward = 0 (DDPG) or -1 (DDPG+HER)*
  - **end**
- **end**
- **if** *Continuous Reward* **then**
  - *Reward = 1 -* $\frac{ComputedDistance \times 0.5}{MaximumDistancewithinWorkspace}$
- **end**
- **return** *Reward*

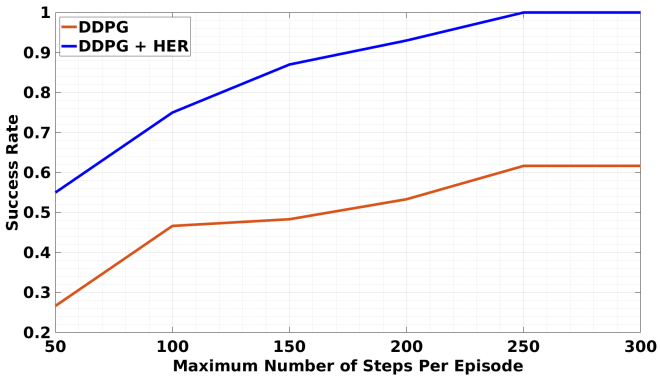


Fig. 5: Success Rate of the DDPG and DDPG+HER Models for dVRK PSM Reach Task with different Maximum Number of Steps per Episode (Average of 3 Iterations). Each iteration consisted of 20 trials to reach the goal position.

B. Results

The success rate was defined as the number of times the dVRK PSM was able to reach the goal state out of 20 trials. Fig. 5 shows the success rate of the two models across different maximum number of steps allowed per episode. The models were trained for 4 millions steps and an aggregate of 3 iterations was used to calculate the success rate. The following observations were made:

DDPG: Achieved a maximum success rate of 61% when the maximum of number of steps allowed per episode was varied between 50-300.

DDPG+HER: Achieved a maximum success rate of 100% when the maximum number of steps allowed per episode was greater than or equal to 250.

The dVRK PSM was successfully able to reach the desired position with a success rate of 100% for episodes which allow 250 or greater steps and goal error margin of 10*mm* while using DDPG+HER model.

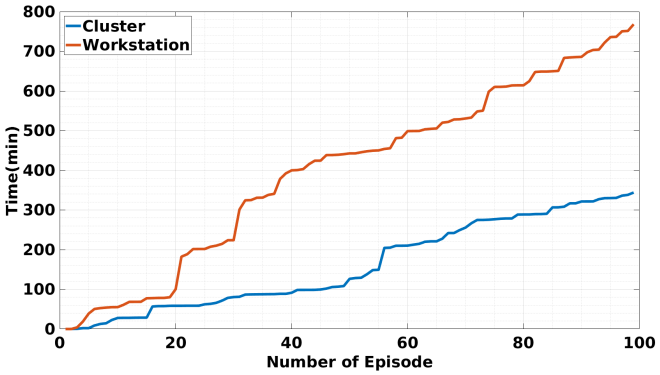


Fig. 6: Time per episode in minutes vs the number of episodes while comparing training on the cluster[3] vs training on a workstation[4]

The above experiments were carried out on a workstation. To validate the performance increase obtained by deploying ARL on the cluster, we measured the time taken to complete **100** episodes. Fig. 6 shows the increased performance obtained by running the training phase on the cluster. Both profiles were run on the same computer with the same background and foreground processes running.

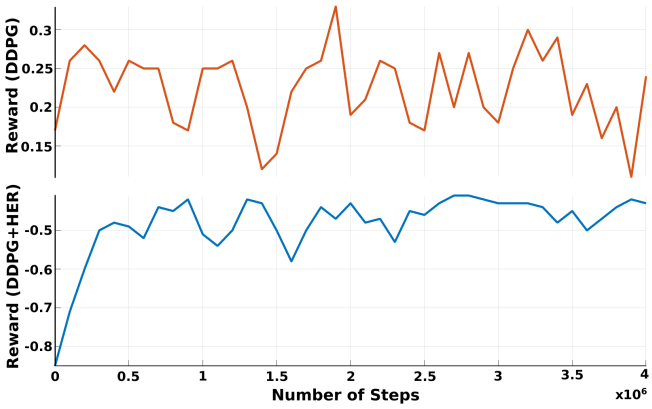


Fig. 7: Rewards received by dVRK PSM over 4 million steps during (a) DDPG (orange) and (b) DDPG+HER (blue) models training.

V. Discussion

From the observations made in the previous section, it can be noted that the model trained using DDPG+HER performed better than the DDPG model. Fig. 7 describes the progression of rewards received by the agent during the training process for DDPG (Fig. 7a), and DDPG+HER (Fig. 7b) respectively. It can be seen that the rewards received by the agent increased significantly after few thousands steps for DDPG+HER, whereas the DDPG model received fluctuating rewards throughout training resulting in improper training. We believe that this could be because of the instability of DDPG due to hyper-parameter sensitivity [39]. We also experienced difficulties while training new environments due to deceptive gradients [40], [41]. Other factors that could have affected training included selection of appropriate noise which can cause inefficient exploration of states [42]. After obtaining optimal policies, we observed that the dVRK PSM performed the desired motion of reaching the goal position required for debris removal successfully. The seamless transfer of the trained policy from simulation to the physical robot shows ARL's flexibility to help researchers test their novel techniques without needing access to the physical robots.

To conclude, we have presented ARL: a real-time simulation based RL toolkit that is tailored for medical robots. The toolkit was validated on the dVRK PSM by performing a reach task which portrayed debris removal using the model trained with DDPG+HER technique. ARL's ability to leverage HPC by running on containerized versions makes it suitable for learning techniques. The added benefits of running a headless instance allows it to be compatible with a larger variety of servers, including window-less ones.

‘

VI. Future Work

The added capability of simulating tissue dynamics using AMBF reduces the complexity of automating tissue-specific tasks using RL techniques for future applications. Furthermore, a database can be created to store successful hyper-parameter values for trained models, easing the adoption of ARL amongst new users. Future experiments can also leverage the containerization and synchronous training capabilities of ARL to train multiple instances of the same task to improve performance by utilizing parameter noise for exploration [43] and explore newer algorithms like the Twin Delayed DDPG (TD3) [44]. The expert data collected from users performing surgical tasks could be used to pre-train RL algorithms [35] using the learning from demonstration module present in ARL. ARL also offers the opportunity to explore new frontiers of RL based techniques to automate medical tasks such as controlling exoskeletons, ablating cancerous tissue, etc. through the variety of medical robot models it has to offer.

VII. Acknowledgement

This work is supported by the National Science Foundation (NSF) through National Robotics Initiative (NRI) grant: IIS-1637759 and NSF AccelNet grant-1927275. The research work involved using computational resources supported by the Academic and Research Computing group at Worcester Polytechnic Institute.

References

[1] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv:1606.01540 [cs]*, Jun. 2016, arXiv: 1606.01540. [Online]. Available: http://arxiv.org/abs/1606.01540

[3] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," 2017. [Online]. Available: https://github.com/openai/baselines

[4] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2018.8460528

[5] B. Osiński, A. Jakubowski, P. Ziecina, P. Miłoś, C. Galias, S. Homoceanu, and H. Michalewski, "Simulation-based reinforcement learning for real-world autonomous driving," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6411–6418.

[6] M. Yip and N. Das, "Robot Autonomy for Surgery," *arXiv:1707.03080 [cs]*, Jul. 2017, arXiv: 1707.03080. [Online]. Available: http://arxiv.org/abs/1707.03080

[7] Y. Kassahun, B. Yu, A. T. Tibebu, S. Giannarou, J. H. Metzen, and V. Poorten, "Surgical Robotics Beyond Enhanced Dexterity Instrumentation," *International journal of computer assisted radiology and surgery*, p. 15. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/26450107/

[8] M. Selvaggio, A. M. G. E, R. Moccia, F. Ficuciello, and B. Siciliano, "Haptic-guided shared control for needle grasping optimization in minimally invasive robotic surgery," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3617–3623.

[9] S. Speidel, A. Kroehnert, S. Bodenstedt, H. Kenngott, B. Müller-Stich, and R. Dillmann, "Image-based tracking of the suturing needle during laparoscopic interventions," R. J. Webster and Z. R. Yaniv, Eds., Orlando, Florida, United States, Mar. 2015, p. 94150B. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2081920

[10] G. Dulan, R. V. Rege, D. C. Hogg, K. M. Gilberg-Fisher, N. A. Arain, S. T. Tesfay, and D. J. Scott, "Developing a comprehensive, proficiency-based training program for robotic surgery," *Surgery*, vol. 152, no. 3, pp. 477–488, Sep. 2012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0039606012003893

[11] 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)*, 2019, pp. 1875–1882.

[12] A. Munawar, N. Srishankar, and G. S. Fischer, "An open-source framework for rapid development of interactive soft-body simulations for real-time training," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, ser. 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE.

[13] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An Open-Source Research Kit for the da Vinci R Surgical System," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, p. 6.

[14] B. Hannaford, J. Rosen, D. W. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. N. Kosari, and L. White, "Raven-ii: An open platform for surgical robotics research," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 4, pp. 954–959, 2013.

[15] C. J. Nycz, R. Gondokaryono, P. Carvalho, N. Patel, M. Wartenberg, J. G. Pilitsis, and G. S. Fischer, "Mechanical validation of an mri compatible stereotactic neurosurgery robot in preparation for pre-clinical trials," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1677–1684.

[16] V. M. Varier, D. K. Rajamani, N. Goldfarb, F. Tavakkolmoghaddam, A. Munawar, and G. S. Fischer, "Collaborative suturing: A reinforcement learning approach to automate hand-off task in suturing for surgical robots," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2020, pp. 1380–1386.

[17] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming Exploration in Reinforcement Learning with Demonstrations," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299, 2018. [Online]. Available: http://arxiv.org/abs/1709.10089

[18] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2016. [Online]. Available: http://arxiv.org/abs/1509.02971

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

[20] N. G. Lopez, Y. L. E. Nuin, E. B. Moral, L. U. S. Juan, A. S. Rueda, V. M. Vilches, and R. Kojcev, "gym-gazebo2, a toolkit for reinforcement learning using ROS 2 and gazebo," *CoRR*, vol. abs/1903.06278, 2019. [Online]. Available: http://arxiv.org/abs/1903.06278

[21] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4243–4250.

[22] B. Balaji, S. Mallya, S. Genc, S. Gupta, L. Dirac, V. Khare, G. Roy, T. Sun, Y. Tao, B. Townsend, E. Calleja, S. Muralidhara, and D. Karuppasamy, "Deepracer: Educational autonomous racing platform for experimentation with sim2real reinforcement learning," *CoRR*, 2019. [Online]. Available: https://arxiv.org/abs/1911.01562

[23] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913495721

[24] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *CoRR*, 2018. [Online]. Available: https://arxiv.org/abs/1801.08757

[25] T. Nguyen, N. D. Nguyen, F. Bello, and S. Nahavandi, "A new tensioning method using deep reinforcement learning for surgical pattern cutting," in *2019 IEEE International Conference on Industrial Technology (ICIT)*, 2019, pp. 1339–1344.

[26] F. Richter, R. K. Orosco, and M. C. Yip, "Open-Sourced Reinforcement Learning Environments for Surgical Robotics," *CoRR*, Jan. 2020, arXiv: 1903.02090. [Online]. Available: http://arxiv.org/abs/1903.02090

[27] X. Tan, C. Chng, Y. Su, K. Lim, and C. Chui, "Robot-assisted training in laparoscopy using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 485–492, 2019.

[28] D. Baek, M. Hwang, H. Kim, and D. Kwon, "Path planning for automation of surgery robot based on probabilistic roadmap and reinforcement learning," in *2018 15th International Conference on Ubiquitous Robots (UR)*, 2018, pp. 342–347.

[29] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013, www.coppeliarobotics.com.

[30] B. Thananjeyan, A. Garg, S. Krishnan, C. Chen, L. Miller, and K. Goldberg, "Multilateral surgical pattern cutting in 2D orthotropic gauze with deep reinforcement learning policies for tensioning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore: IEEE, May 2017, pp. 2371–2378. [Online]. Available: http://ieeexplore.ieee.org/document/7989275/

[31] J. Xu, B. Li, B. Lu, Y.-H. Liu, Q. Dou, and P.-A. Heng, "Surrol: An open-source reinforcement learning centered and dvrk compatible platform for surgical robot learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1821–1828.

[32] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation in robotics, games and machine learning," 2017. [Online]. Available: http://pybullet.org

[33] F. Conti, F. Barbagli, R. Balaniuk, M. Halg, C. Lu, D. Morris, L. Sentis, J. Warren, O. Khatib, and K. Salisbury, "The chai libraries," in *Proceedings of Eurohaptics 2003*, Dublin, Ireland, 2003, pp. 496–500.

[34] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," *ICRA Workshop on Open Source Software*, vol. 3, 01 2009. [Online]. Available: https://www.researchgate.net/publication/233881999_ROS_an_open-source_Robot_Operating_System

[35] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," *Advances in neural information processing systems*, vol. 29, 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/hash/cc7e2b878868cbae992d1fb743995d8f-Abstract.html

[36] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[37] Z. Chen, A. Deguet, R. Taylor, S. DiMaio, G. Fischer, and P. Kazanzides, "An Open-Source Hardware and Software Platform for Telesurgical Robotics Research," *The MIDAS Journal*, p. 11. [Online]. Available: https://www.midasjournal.org/browse/publication/892

[38] Z. Chen, A. Deguet, R. H. Taylor, and P. Kazanzides, "Software Architecture of the Da Vinci Research Kit," in *2017 First IEEE International Conference on Robotic Computing (IRC)*. Taichung, Taiwan: IEEE, Apr. 2017, pp. 180–187. [Online]. Available: http://ieeexplore.ieee.org/document/7926536/

[39] G. Matheron, N. Perrin, and O. Sigaud, "The problem with ddpg: understanding failures in deterministic environments with sparse rewards," *CoRR*, vol. abs/1911.11679, 2019. [Online]. Available: https://openreview.net/forum?id=HyxnH64KwS

[40] C. Colas, O. Sigaud, and P.-Y. Oudeyer, "GEP-PG: Decoupling Exploration and Exploitation in Deep Reinforcement Learning Algorithms," *International conference on machine learning*, pp. 1039–1048, 2018. [Online]. Available: https://proceedings.mlr.press/v80/colas18a.html

[41] R. Liessner., J. Schmitt., A. Dietermann., and B. Bäker., "Hyperparameter optimization for deep reinforcement learning in vehicle energy management," in *Proceedings of the 11th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, INSTICC. SciTePress, 2019, pp. 134–144.

[42] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy Networks for Exploration," *CoRR*, Jul. 2019, arXiv: 1706.10295. [Online]. Available: http://arxiv.org/abs/1706.10295

[43] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter Space Noise for Exploration," *CoRR*, Jan. 2018, arXiv: 1706.01905. [Online]. Available: http://arxiv.org/abs/1706.01905

[44] S. Fujimoto, H. v. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *International conference on machine learning*, pp. 1587–1596, 2018. [Online]. Available: https://proceedings.mlr.press/v80/fujimoto18a.html