Robotic Embodiment of Human-Like Motor Skills via Reinforcement Learning

Luis Guzman[1], Vassilios Morellas[2], and Nikolaos Papanikolopoulos[1]

***Abstract*—Current methods require robots to be reprogrammed for every new task, consuming many engineering resources. This work focuses on integrating real and simulated environments for our proposed "Internet of Skills," which enables robots to learn advanced skills from a small set of expert demonstrations. By expanding on recent work in the areas of Learning from Demonstrations (LfD) and Reinforcement Learning (RL), we can train robot control policies that can not only effectively complete a given task but also do so with greater performance than the expert demonstrations used to train the policy. In this work, we create simulated environments to train RL algorithms for the task of inverse kinematics and obstacle avoidance. Many state-of-the-art RL algorithms are compared, and we provide a detailed analysis of the state space and parameters chosen. Lastly, we utilize a Vicon motion tracking system and train the robot agent to follow trajectories given by a human operator. Our results show that reinforcement learning algorithms such as proximal policy optimization can develop control policies that are capable of complex control tasks that integrate with the real world, an important first step towards developing a system that can autonomously learn new skills from human demonstrations.**

Telerobotics and Teleoperation, Reinforcement Learning, Transfer Learning, Model Learning for Control, Collision Avoidance

I. Introduction

STATE-of-the-art Reinforcement Learning (RL) algorithms have partially addressed hard exploration problems through greater sample efficiency and reducing approximation error [1], but many still struggle when goals are abstract or significant future planning is necessary. To address these challenges, Learning from Demonstrations (LfD) emerged as a means of determining an agent's policy through observation of an expert (human) demonstrator [2]. This form of learning integrates human and machine capabilities to complement each other in carrying out complex tasks that are too hazardous and difficult to be performed solely by humans or robots. Our proposed Internet of Skills (IoS) enables practical applications of this research by utilizing motion capture technology in order to record expert demonstrations. In this paper, we build the foundations of this system by training RL control policies for a hand following and obstacle avoidance task. We implement this policy on a Kinova Gen3 Robot, demonstrating a working sim-to-real prototype that replicates human motion on a physical robot. This research will enable future methods which use human demonstrations to teach robots to complete new skills autonomously.

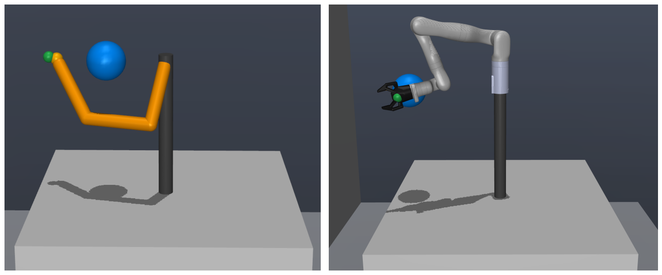

(a) 5 DOF robot (b) 7 DOF Kinova Gen3 Robot

Fig. 1: Example MuJoCo virtual environments. The target point is shown in green, and the obstacle is in blue.

Currently, training a robot to perform a new task usually requires some form of teleoperation [3], where an operator is required to either set robot joint angles directly by manipulating a custom controller or set end-effector coordinates that can then be used to solve robot joint angles through inverse kinematics. Both of these methods are specific to a given robot's actuator layout (phenotype), and often an entirely new teleoperation system must be designed to control a robot with a different configuration. Additionally, having the operator perform a task using counter-intuitive human-machine interfaces lacks the fluidity and nuance of certain gestures and requires the operator to become accustomed to the teleoperation system. For these reasons, a system that allows human operators to seamlessly integrate and control robots of any phenotype is desired.

Forming this connection between human and robot actions requires a shared language where human-given commands can understood by any robot; however, human action is difficult to represent in an analytic form. Learning from demonstrations is a technique for determining a robot agent's control policy from human (expert) demonstrations. It shares similarity with reinforcement learning, although in RL, a policy is learned through interaction and exploration in a training environment rather than from expert demonstrations. In both RL and LfD, the robot's surroundings are represented by a state S and the robot performs actions A. The same state and action space can be used to quantify the experience of a human demonstrator. A

Manuscript received: September 9, 2021; Revised December 7, 2021; Accepted January 20, 2022.

This paper was recommended for publication by Editor Jee-Hwan Ryu upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Minnesota Robotics Institute (MnRI), Honeywell, the National Science Foundation, and USDA/NIFA.

[1]Luis Guzman and Nikolaos Papanikolopoulos are with the Department of Computer Science and Engineering, University of Minnesota, { guzma102 | papan001 }@umn.edu

[2] Vassilios Morellas is with the Department of Electrical and Computer Engineering, University of Minnesota. morellas@umn.edu

Digital Object Identifier (DOI): see top of this page.

policy π maps the robot's observed surroundings to an action $\pi : S \rightarrow A$. The goal is to learn a policy that either optimizes a certain reward function (RL) or one that performs similar actions to an expert demonstrator (LfD).

Although this paper focuses on reinforcement learning, it is important to note the benefits of using demonstrations, since it provides context into the larger goals of this project. Using expert demonstrations has been shown to improve convergence rates and performance, especially in high-dimensional tasks such as object manipulation [2]. Furthermore, using reinforcement learning algorithms simultaneously with expert demonstrations can produce a policy that exceeds the performance of "imperfect" expert demonstrations [4]–[7]. Possible benefits include higher precision than humanly-attainable and the filtering of hand-shaking when performing delicate manipulation tasks. In this paper, we provide detailed analysis of reinforcement learning without demonstrations on the tasks of inverse kinematics and obstacle avoidance, which will inform future publications that utilize LfD.

We draw inspiration from Sangiovanni et al. [8], which uses reinforcement learning to solve inverse kinematics and obstacle avoidance. Particularly, our choice of reward function is inspired by this work. They use a method known as normalized advantage functions (NAF), which is similar to deep Q-learning for continuous tasks. We utilize more recent RL algorithms that have been shown to have better convergence rates and performance than NAF. Furthermore, we propose significant modifications to the method, which improves agent performance and stability, and we implement the policy on a physical Kinova Gen3 robot.

Proximal Policy Optimization (PPO) [9] is an example of an on-policy reinforcement learning algorithm. It is characterized by a policy update equation that includes a clipping parameter, which ensures that no single policy update is too large and destroys the policy. Previous work has shown success when applying PPO to robot control tasks, such as object manipulation [10] and control of a humanoid robot [11]. PPO is robust to hyperparameters, so it requires very little tuning in order to get optimal performance.

Twin delayed deep deterministic policy gradients (TD3) [1] is a modified version of the deep deterministic policy gradients (DDPG) [12] algorithm. It uses two critic networks in order to minimize the Q-function approximation error, and they delay the policy updates to every-other update of the critic network to further decrease the error. It features a higher sample efficiency than PPO since it is an off-policy algorithm and thus stores its experience in a replay buffer so that preferable trajectories can be reused for training. In general, off-policy algorithms are less stable than on-policy ones, so more hyperparameter tuning may be required.

Actor-critic using Kronecker-factored trust region (ACKTR) [13] and advantage actor-critic (A2C) [14] are two other examples of on-policy learning algorithms, and soft actor-critic (SAC) [15] and deep deterministic policy gradients (DDPG) [12] are additional off-policy algorithms that will appear later in this paper.

Through the combined capability of reinforcement learning and learning from demonstrations, robots will be able to quickly acquire new skills and perform tasks in uncertain environments. In this paper, we train RL algorithms for a hand following and obstacle avoidance task. Our results show that our choice of observation space and reward function improves convergence times and increases performance by 27.3% when compared to the method of Sangiovanni et al. [8] on the same task. We demonstrate practical applications of this research by integrating the virtual environments with a Vicon motion capture system, which enables an operator to control a Kinova robot using only the movements of their own body. This coupling of human and robot action will enable our proposed Internet of Skills, which enables robots to learn new skills autonomously from observing expert demonstrations.

II. Method

A. Virtual Environments

OpenAI Gym [16] is an open-source toolkit for training RL algorithms within virtual environments. We built the virtual environments within this framework in order to provide a familiar interface that can be reused when testing multiple RL and LfD algorithms. OpenAI gym also supports the MuJoCo physics engine, which we will use for its superior simulation accuracy, including electro-mechanical responses of the robot actuators. MuJoCo is capable of using the Unified Robot Description Format (URDF) so that a multitude of robot configurations can easily be loaded into the simulation.

For initial testing, we created two simulated robots to be used in the Gym environments. The first robot is a 5 degree of freedom (DOF) manipulator, shown in Figure 1a. This robot is a modified version of the OpenAI Reacher environment and is a simplified model of a human arm, with a 3 DOF ball joint at the shoulder, and two single DOF hinge joints at the elbow and wrist. The second robot is a 7 DOF Kinova Gen3 Robot, shown in Figure 1b. Kinova supplies URDF models for their robots, and these have been converted to the MuJoCo XML format by the community [17].

Each environment features an obstacle (in blue), and a target point (shown in green). The robot must learn to reach the target point, while avoiding the obstacle. This is a non-trivial task, since every goal location has infinitely many possible solutions for the joint angles of the robot.

B. Reward Function

For our application, we build on the reward function presented in Sangiovanni et al. [8].

$$r = c_1 R_T + c_2 R_A + c_3 R_O + c_4 R_X.$$

Large distances from the end effector to the target point are penalized with the R_T term. Sangiovanni et al. propose a Huber Loss function for this purpose.

$$R_T = L_\delta(d) = -\begin{cases} \frac{1}{2}d^2 & \text{for } |d| < \delta \\ \delta(|d| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$$

where d is the Euclidean distance between the target point and the end effector and δ is the Huber Loss parameter, which determines the regions of linear and quadratic loss.

R_A penalizes large actions (a) to reduce overshoot and encourage the manipulator to remain stationary after it has reached the target point.

$$R_A = -||a||^2.$$

During training, we disable collisions in the MuJoCo environment so that the manipulator can pass freely through the obstacles. This is to ensure continuity of the action space —with collisions enabled, a large action could result in zero movement, which would cause an incorrect policy update. If the manipulator passes through an obstacle, it receives a large negative reward in the form of R_O.

$$R_O = -\left(\frac{d_{\text{ref}}}{d_O + d_{\text{ref}}}\right)^p$$

where d_O is the minimum distance from the manipulator to the center of obstacle, and p sets the exponential decay rate. d_{ref} is a constant that determines how close the manipulator can get to the obstacle without incurring significant penalty. This value is set to be approximately the size of the obstacle or slightly larger.

Because collisions are disabled in MuJoCo, we also penalize behavior which would be impossible on a physical robot. R_X discourages behavior where the manipulator passes through itself to reach the target point by penalizing large joint angles.

$$R_X = \sum_{n=0}^{N} \min\left(0, \frac{\pi}{2} - \theta_n\right)$$

where N is the total number of hinge joints and θ_n is the angle of each joint.

C. Observation & Action Spaces

The observation space defines what information the robot agent can access to determine its policy. At each timestep, the robot agent receives:

$$\{\ q,\ \dot{q},\ q_T,\ q_O,\ d_T,\ d_O\ \}$$

where q and $\dot{q}$ represent the robot's joint angles and velocities, q_T is the location of the target, and q_O is the location of the obstacle. Joint angles are represented as cosine/sine pairs in order to stabilize training and improve performance (i.e. $q_i \to (\cos(q_i), \sin(q_i))$. For ball joints that use quaternions, raw quaternion values are used. d_T and d_O are the distances from the robot to the target and obstacle, respectively. These last two values were added to improve convergence times, as these are the actual values that we would like the agent to optimize. Additional discussion and data that supports this choice of observation space are provided in the results.

In the case of a ball-type joint, the observation space gains an additional dimension, since ball joints are represented as 4 dimensional quaternions. This has the disadvantage of adding additional dimensionality that can slow down training, but it crucially avoids the problem of gimbal lock, which can cause exploding gradients during training.

The action space is comparatively simple, and is equal to the number of degrees of freedom of the robot. The network is trained to output motor control commands, so all error correction must be learned by the policy network.

Optimizing over this state space is a challenging task. For the 5 DOF robot with a ball-type shoulder joint, the observation space includes 20 continuous dimensions (6 joint angles, 6 joint velocities, 3 target coordinates, 3 obstacle coordinates, and 2 distances), and the action space includes 5 continuous dimensions. Since attempting to explore the entire state space would be impossible, we must utilize state-of-the-art reinforcement learning algorithms and techniques to simplify this exploration problem.

D. Training

In order to train the agent to complete complex tasks, we segment the training process into easier sub-tasks. In the first task, we ask the agent to reach stationary goal points that are near the obstacle. For high DOF robots, we can further limit this first sub-task to the $z = 0$ plane, and include the z dimension later. By placing this constraint, we decrease the size of the observation space to explore, so training is quicker and more likely to converge.

Next, through a process called *experience copy*, we load in the policy we trained on the first sub-task, and begin the training process on the next sub-task. The second sub-task is to trace out a variety of arc patterns. Ideally, all points that make-up the arc trajectory have been seen by the agent during the training of the first sub-task. Training on the second task makes minor improvements to the agent's ability to track moving objects.

All policy networks were trained using a machine running Ubuntu 20.04 with an Intel(R) Core(TM) i7-4770K CPU @ 3.90GHz, 16 GiB of memory, and a Nvidia GeForce RTX 3070 GPU. Initially, policy networks were trained to 10 million time steps, but we noticed very little improvement after 5 million steps. All figures and results are reported at 5 million time steps. In order to minimize the effect of stochastic policy training, networks were trained three times using different seed values. The reported results are the average of the three training instances.

For the implementations of PPO, ACKTR, and A2C, we modified the implementation of [18]. For TD3 and DDPG,

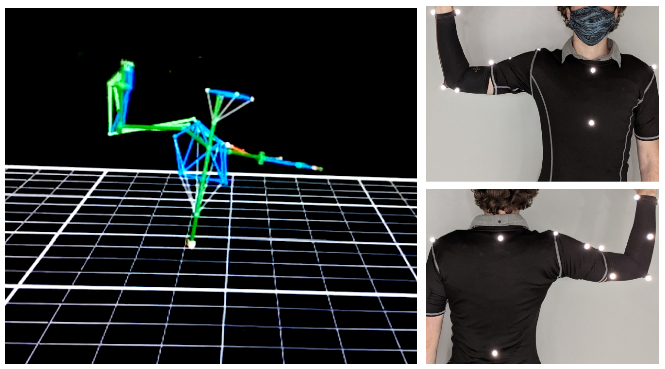
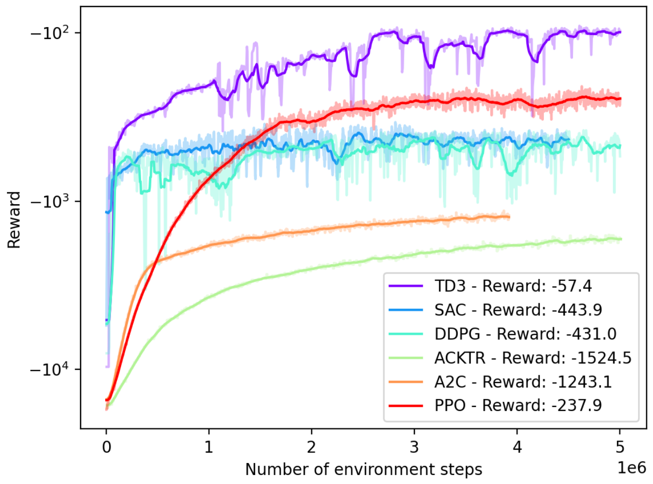
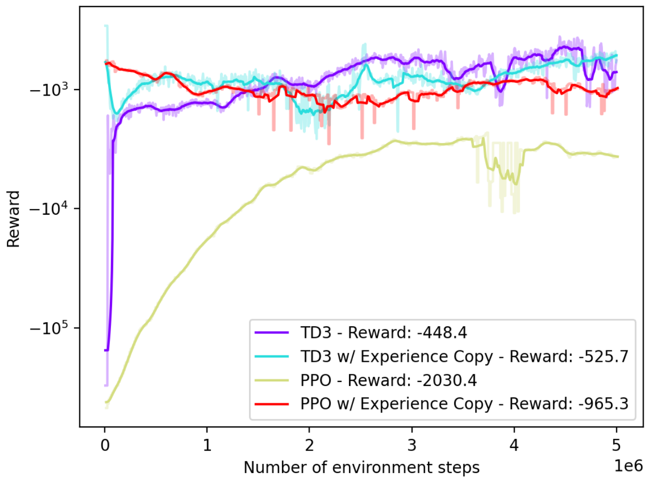


Fig. 2: View of the motion tracking data and tracker placement. The expert's entire right arm and torso are tracked, although only the hand position is used in this paper.



(a) Comparison of RL algorithms on the stationary goal environment



(b) Comparison of PPO and TD3 experience copy on the trajectory following environment

Fig. 3: Results on two of the simulated virtual environments.

we used [1]. For SAC, we used [19]. All networks are implemented using PyTorch.

E. Experimental setup

As described in the previous section, the first experiment tasks the robot with reaching a random goal point, while also avoiding an obstacle in the workspace. Since locations are randomized, all simulation and evaluation occurs within the MuJoCo virtual environment.

The next case we explore is teleoperation. In this experiment, a human operator gives the agent a target trajectory to follow, and the agent is tasked to decide the optimal way of reaching that goal. Expert trajectories are recorded using a Vicon motion tracking system and reflective markers, which enable localization of the expert's arm. In this paper, we only use the position of the expert's hand, and utilizing the full arm tracking data is left to future work. An example image of the tracker placement and motion tracking data is shown in Figure 2. The dimensions of the motion tracking space are aligned with the dimensions of the MuJoCo environment by normalizing the maximum workspace coordinates of each environment.

We ran the majority of experiments on the 5 DOF robot from Figure 1a. We also implemented the PPO trajectory following policy on a physical Kinova Gen3 robot to demonstrate the versatility of this approach. The Kinova robot requires 1 kHz control feedback in order to prevent jerky motion, so low-level control must be handled by the embedded device on-board the Kinova robot [20]. Our control interface sends the robot joint velocities, which are proportional to the error between the desired and current joint angles. The joint velocity commands and joint feedback are communicated using Kinova's Robot Operating System (ROS) interface.

III. Results

The results of six different RL algorithms on the stationary goal environment are shown in Figure 3a. Here we can see that TD3 and PPO were the top performing algorithms, with SAC and DDPG tied for third. We observe that the off-policy algorithms (TD3, SAC, and DDPG) converged very quickly, then made incremental improvements for the remainder of the training time. In contrast, the on-policy algorithms took much longer to converge but they experienced a more stable training process (i.e., the reward value nearly monotonically increases). Since these values are averaged over three trials, TD3 and DDPG appear to be more stable than their single trials, where we often observed reward fluctuations up to an order of magnitude in size.

In Figure 3b we compare the performance of PPO and TD3 when using experience copy. The copied policies are compared against policies that are trained directly on the trajectory following environment. In this test, the sample efficiency of TD3 is particularly noticeable. Any advantage that is gained by using experience copy is quickly negated by "vanilla" TD3's fast convergence to near-optimal values. PPO, on the other hand, showed a 26% performance gain when using experience copy. The additional reward was enough to achieve comparable performance to TD3 on the trajectory-following task.

When deciding what algorithm to apply for future experiments, we must consider more than just the asymptotic reward value. The PPO implementation is parallelized over 8 environment instances, so PPO took 45 minutes to train, whereas TD3 was single-threaded and took over 30 hours to train for the same number of time steps. However, one advantage of TD3 is its sample efficiency, and as shown in Figure 3a, it reaches its maximum reward at around two million time steps. The corresponding training training time is 7.5 hours, which is still much longer than it takes for PPO to converge.

Additionally, when viewing the control policies in simulation, we noticed that TD3 produced irregular trajectories, whereas PPO tended to follow smooth arcs. This is quantified in Figure 4, where we compare the distances to the target of the two policies. Although TD3 accrues a better reward, PPO

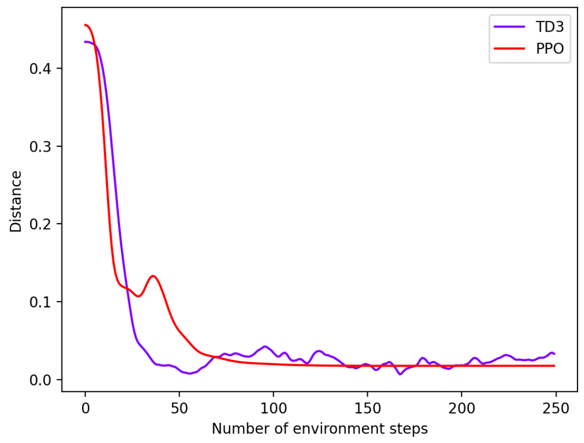
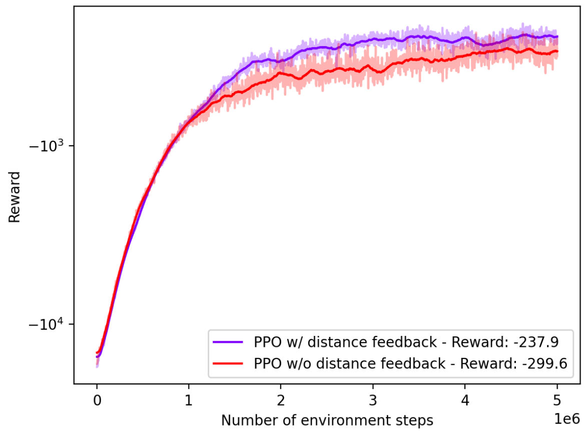
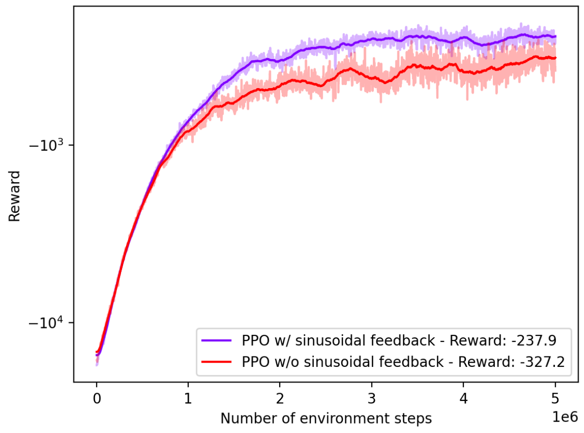


Fig. 4: Distances between the robot fingertip and the target for both the PPO and TD3 policy networks. PPO is found to be much more stable, despite having a lower average reward than TD3.



(a) Reward comparison of the additional distance information



(b) Reward comparison of the sinusoidal joint angles

Fig. 5: Comparison showing performance increase of two of the proposed modifications to the observation space.

shows preferable behavior by holding the fingertip stationary at the target point. This gives major preference to choosing PPO for manipulator control tasks due to its comparably safe and predicable trajectories.

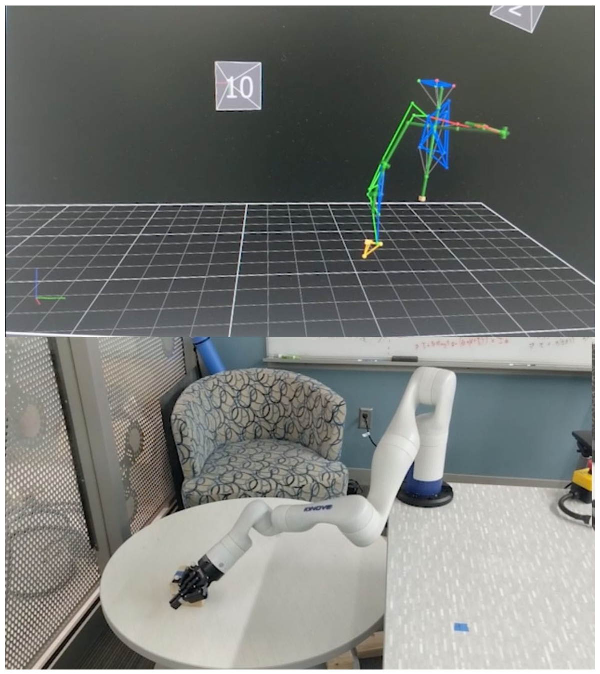


Fig. 6: Demonstration of the PPO hand following policy on a Kinova Gen3 Robot. Hand location of the expert (top) is replicated on the Kinova robot (bottom).

In Figure 5, we provide some experiments that demonstrate the performance gains of the method described in Section II-C. Figure 5a considers the inclusion of the additional distance information (d_T and d_O) in the observation space. Including this additional information showed a 20.6% increase in the asymptotic reward. Similarly, Figure 5b shows that applying the sinusoidal transformation to the raw joint angles improved performance by 27.3%. These improvements to the method of Sangiovanni et al. [8] can be applied to future LfD tasks to improve stability and performance.

An initial test of our sim-to-real pipeline is shown in Figure 6. The robot correctly follows the location of the expert's hand while avoiding obstacles in the workspace. In the current prototype, no external sensors were added to the robot, so we hard-coded the obstacle locations using the state space presented in section II-C. Our results demonstrate a successful transfer of the trajectory following policy, enabling the human operator to control the Kinova robot using only the movements of their own body.

In the appendix, Figure 7 shows a comparison of various values of the environment hyperparameters. Since changing these values affects the calculated reward of a given policy, we chose to evaluate them on average distances to the obstacle and the target. A small target distance and large obstacle distance are desired. We note that c_1 and c_3 had the smallest impact on performance simply because the magnitude of R_T and R_O is smaller than the other values. After training the control policy using the optimal hyperparameters, we observed more robust obstacle avoidance, where the average obstacle distance rose from 4.97 cm to 11.83 cm. The target distance was not significantly affected, rising from 7.18 cm to 8.54 cm, which is within the margin of error we recorded for multiple trials.

Robot Policy	Mean Error (cm)
Kinova	15.2
5DOF	11.2

TABLE I: Comparison of target point error between different robots

As shown in Table I, both the 5 DOF and 7 DOF Kinova robots were able to successfully reach the goal points within 13 cm average error. The 5DOF robot performed better than the Kinova robot, likely because the shoulder ball joint offers far more flexibility than the three real shoulder actuators on the Kinova robot. The values in Table I include cases where the goal point is inside of the obstacle, where I observed the robot to correctly avoid the obstacle while maintaining the closest possible distance to the goal point. This is an important feature of this system, since it demonstrates robust obstacle avoidance, even if the human operator is unaware of an obstacle and instructs the robot to collide with it. These results demonstrate a successful method for using real-world trajectories to operate a robot using reinforcement learning-based control policies, a necessary prerequisite for using demonstrations to train robots to complete more complex tasks.

IV. Conclusion

In this work, we created virtual environments that will enable the Internet of Skills framework, and we tested how state of the art reinforcement learning methods perform in the absence of expert demonstrations. We introduce an observation space that improves performance on inverse kinematics and obstacle avoidance tasks and provide comparisons of multiple RL algorithms. Although PPO achieved only second-best reward values, we recommend using it over other RL algorithms due to its predicable and stable trajectories and its fast training times. Additionally, through the use of experience copy, PPO can achieve comparable performance to state of the art methods like TD3. This work provides foundational insight and explores the methods necessary for developing an Internet of Skills that will enable robots to learn complex tasks from human demonstrations.

V. Future work

Future goals for this project are to utilize the expert's full joint angle data from the motion tracking system, rather than just the position of the expert's hand. Doing so would allow the expert and robot to share the same observation and action space, enabling the use of learning from demonstration algorithms such as Generative Adversarial Imitation Learning (GAIL). Utilizing LfD can speed up training times and allow the robot agent to complete more difficult tasks.

We would also like to extensively test the sim-to-real prototype on a variety of manipulation tasks. This data will be important to further demonstrate the feasibility of such a system, and it can be compared to a future control method which follows all joint angles of the expert.

Lastly, we propose the use of recurrent neural networks (RNNs) to form a mapping from human to robot joint angles. Robots that have significantly different actuator layout than a human arm will not be able to directly interpret the actions that the human experts take in the environment. This constraint led us to choosing the Kinova 7DOF robot for our initial testing, since the human-robot mapping is nearly identity. Learning a joint angle mapping with RNNs would allow for any robot phenotype to learn from human demonstrations, even if the number of actuators or physical layout varied greatly from a human's.

Appendix A - Hyperparameters

Parameter	Optimal Value	Estimated Value*
c_1	1500	1000
c_2	10	10
c_3	200	60
c_4	0.01	0.01
δ	0.005	0.01
p	6	8
d_{ref}	0.03	0.03

TABLE II: Environment Hyperparameters.
*Estimated values were used for generating Figures 3-5.

Parameter	Value
Learning rate	$3 * 10^{-4}$
Entropy coefficient	0.01
Value loss coefficient	0.5
Epochs per update	10
Num mini batch	1
Discount Factor (γ)	0.99
GAE Discount Factor (γ_{GAE})	0.95
Clip	0.2

TABLE III: PPO Hyperparameters

Parameter	Value
Start Timesteps	$25 * 10^3$
Evaluation Frequency	$5 * 10^3$
Noise STD	0.1
Batch Size	256
Discount Factor (γ)	0.99
Tau τ	0.005
Policy Frequency	2

TABLE IV: TD3 Hyperparameters

ACKNOWLEDGMENT

The authors would like to thank all the members of the Center for Distributed Robotics Laboratory for their help. This material is based upon work partially supported by the Minnesota Robotics Institute (MnRI), Honeywell, and the National Science Foundation through grants CNS-1439728, CNS-1531330, and CNS-1939033. USDA/NIFA has also supported this work through the grant 2020-67021-30755.

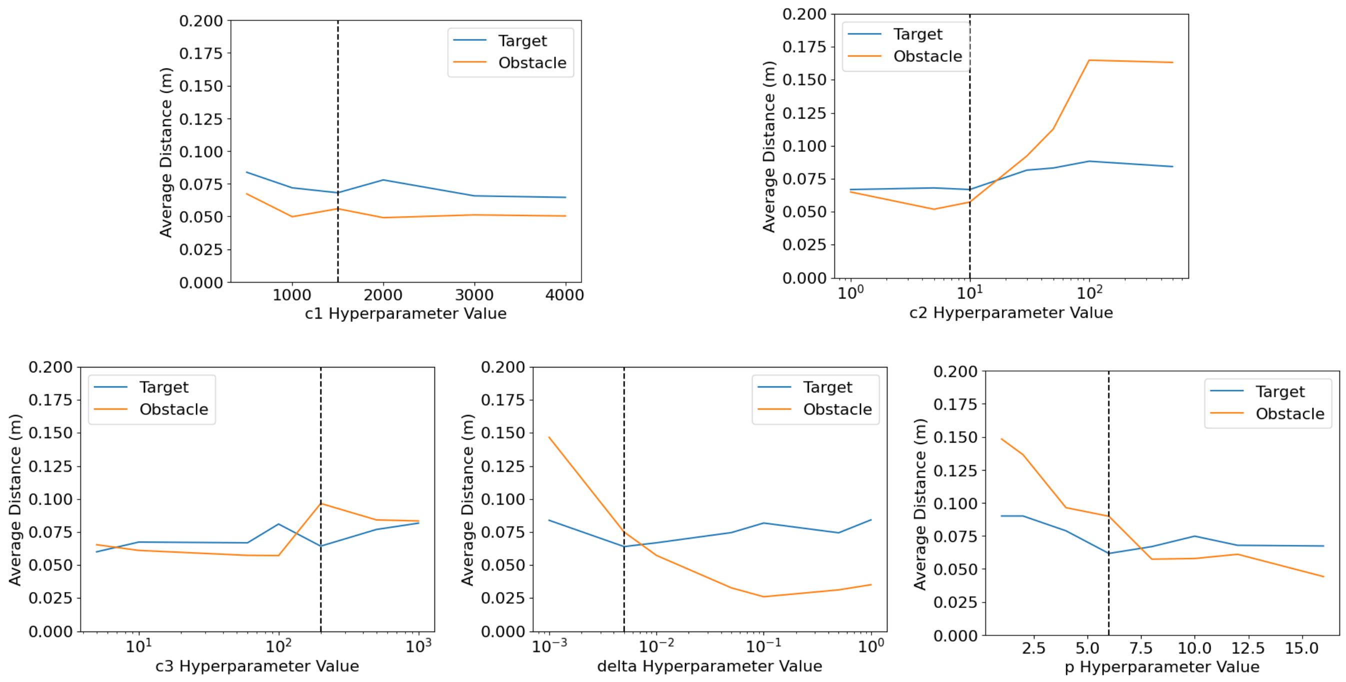


Fig. 7: Performance comparison of various hyperparameter choices for the 5 DOF robot reaching stationary goals. Optimal values are marked with the black dashed line and are given in Table IV. c_4 and d_{ref} are chosen empirically since there is no quantifiable selection method. The same hyperparameters were used for the trajectory following task.

References

[1] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018. [Online]. Available: http://arxiv.org/abs/1802.09477

[2] T. L. Paine, C. Gulcehre, B. Shahriari, M. Denil, M. Hoffman, H. Soyer, R. Tanburn, S. Kapturowski, N. Rabinowitz, D. Williams, G. Barth-Maron, Z. Wang, N. de Freitas, and W. Team, "Making efficient use of demonstrations to solve hard exploration problems," 2019.

[3] S. Hirche and M. Buss, "Human-oriented control for haptic teleoperation," *Proceedings of the IEEE*, vol. 100, no. 3, pp. 623–647, 2012.

[4] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, and N. R. Waytowich, "Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments," 2020.

[5] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys, "Deep q-learning from demonstrations," 2017.

[6] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," *CoRR*, vol. abs/1709.10089, 2017. [Online]. Available: http://arxiv.org/abs/1709.10089

[7] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 2469–2478. [Online]. Available: https://proceedings.mlr.press/v80/kang18a.html

[8] B. Sangiovanni, A. Rendiniello, G. P. Incremona, A. Ferrara, and M. Piastra, "Deep reinforcement learning for collision avoidance of robotic manipulators," in *2018 European Control Conference (ECC)*, 2018, pp. 2063–2068.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[10] A. A. Shahid, L. Roveda, D. Piga, and F. Braghin, "Learning continuous control actions for robotic grasping with reinforcement learning," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 4066–4072.

[11] L. C. Melo, D. C. Melo, and M. R. Maximo, "Learning humanoid robot running motions with symmetry incentive through proximal policy optimization," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 3, pp. 1–15, 2021.

[12] 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.

[13] Y. Wu, E. Mansimov, S. Liao, R. B. Grosse, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," *CoRR*, vol. abs/1708.05144, 2017. [Online]. Available: http://arxiv.org/abs/1708.05144

[14] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016. [Online]. Available: http://arxiv.org/abs/1602.01783

[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *CoRR*, vol. abs/1801.01290, 2018. [Online]. Available: http://arxiv.org/abs/1801.01290

[16] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[17] V. Zhang, "Gen3 mujoco," https://github.com/vincentzhang/gen3-mujoco, 2019.

[18] I. Kostrikov, "Pytorch implementations of reinforcement learning algorithms," https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

[19] R. Yang, "torchrl," https://github.com/RchalYang/torchrl, 2021.

[20] KinovaRobotics, "ros_kortex," https://github.com/Kinovarobotics/ros_kortex, 2021.