Evaluation of Techniques for Sim2Real Reinforcement Learning

Mahesh Ranaweera, Qusay H. Mahmoud
Department of Electrical, Computer and Software Engineering, Ontario Tech University, Oshawa, ON, L1G 0C5 Canada
{mahesh.ranaweerakaluarachchige, qusay.mahmoud}@ontariotechu.net

Abstract

Reinforcement learning (RL) has demonstrated promising results in transferring learned policies from simulation to real-world environments. However, inconsistencies and discrepancies between the two environments cause a negative transfer. The phenomenon is commonly known as the "reality gap." The reality gap prevents learned policies from generalizing to the physical environment. This paper aims to evaluate techniques to improve sim2real learning and bridge the reality gap using RL. For this research, a 3-DOF Stewart Platform was built virtually and physically. The goal of the platform was to guide and balance the marble towards the center of the Stewart platform. Custom API was created to induce noise, manipulate in-game physics, dynamics, and lighting conditions, and perform domain randomization to improve generalization. Two RL algorithms; Q-Learning and Actor-Critic were implemented to train the agent and to evaluate the performance in bridging the reality gap. This paper outlines the techniques utilized to create noise, domain randomization, perform training, results, and observations. Overall, the obtained results show the effectiveness of domain randomization and inducing noise during the agents' learning process. Additionally, the findings provide valuable insights into implementing sim2real RL algorithms to bridge the reality gap.

Introduction

Advanced robotic systems have become the workforce of the future. These platforms have been developed and researched to improve the day-to-day life of humans and for the advancement of the industrial, transportation, healthcare, and exploration sectors. However, creating robots that have the dexterity and capability of humans or animals, or that can effectively mimic their behavior, presents a significant challenge. Since the real-world environment is dynamic in nature, creating advanced robotic platforms is often difficult; therefore, modern robotic systems are deployed in controlled environments such as warehouses that utilize advanced sensors. Deployed robots are pre-programmed to perform tasks in the given domain. Recent developments in machine learning and machine learning techniques are currently being used to train robots to perform advanced tasks.

One of the major disadvantages of machine learning is the requirement for large quantities of quality training data for the training. For robotic applications, vast amounts of handcrafted, generated, or real data are required. However, acquiring a large amount of quality training data is often expensive, dangerous, and time-consuming. Even if the data are acquired, deploying on an unseen domain will require retraining the model with the new data. Without quality training data, the trained model will either be biased or may lack the expected outcomes.

One of the emerging research areas is virtual-to-real transfer learning which utilizes a virtual environment to train the model. Virtual environments allow researchers to create a vast amount of training data, simulating environment conditions that are difficult, dangerous, or expensive to simulate in the real world. Although there are many benefits of using a virtual environment, a virtual environment cannot capture the real-world dynamics and collisions (OpenAI et al. 2019). Real-world dynamics such as gravity, friction, and surface collisions are complex to be modeled or simulated in a virtual environment. In most cases, it requires higher computation power to simulate the virtual environments. Noise, simulated sensor disparity, calibration errors, gear backslash, environment factors, textures, reflection, refraction, and latency are some of the factors that might affect the performance of the model after transferring to a real-world environment. The performance of the model is affected by the inherent differences between the virtual and physical environments, this phenomenon is known as the reality gap (Lomnitz et al. 2020).

This paper is an extension of our previous work (Ranaweera and Mahmoud 2023; 2022) by enhancing the model's performance and generalizability. A 3-DOF Stewart Platform was built virtually and physically for making observations and performing training. The virtual platform allows the training of the RL agent in the simulated environment. This allows further configuration and control of the simulated environment attributes. The goal for the RL agent is to guide the marble toward the center of the Stewart platform. This requires precise control over the servos while utilizing external sensors to make observations. Two RL algorithms; Deep Q-Learning and Actor-Critic were created to train the agent in the virtual environment. To generalize the agent for the physical environment, domain randomiza-

tion, and noise were induced to increase the variability of the training data. The hypothesis is that by increasing the variability of the virtual environment the agent could be seen as one of the variants. This paper evaluates how each randomization affects the overall learning of the agent. The paper also evaluates the agent's success rate in the physical environment for each randomization.

Related Work

Multiple research has been conducted to evaluate different techniques that bridge the reality gap in Sim2Real learning. The goal of using a virtual environment is to generate training data when quality data is unavailable. This methodology is known as Sim2Real transfer learning (Doersch and Zisserman 2019). The learning algorithm performs random actions in the virtual environment to make observations and train the model. However, there is no guarantee that the model trained on the synthetic data can perform and/or adapt to the physical environment. Lipton et al. (Lipson et al. 2006) state that the discrepancy is due to Quasi-static kinematics where the model is unable to accurately predict the dynamics of the physical environment. Since the machine dynamics are sensitive to environmental factors such as noise, variations in parameters, and initial conditions. It is very difficult to create a virtual environment that can accurately model the target physical environment. The researched literature proposes various methods to address the reality gap. Some of the common techniques according to (Ranaweera and Mahmoud 2021) are domain randomization, domain adaptation, generating realistic virtual environments, and developing simplified simulations that focus on capturing essential features.

Domain randomization(DR) is one of the common methods used in Sim2Real research. The main idea of this method is to increase the variability of the training environment by randomly changing virtual environment parameters and attributes. Research (OpenAI et al. 2019), (Tobin et al. 2017), (Park, Kim, and Kim 2020) demonstrate that domain randomization techniques create a rich distribution of training data and variety, allowing the model to only learn the policies and to show that the model can make accurate predictions on the physical environment. OpenAI (OpenAI et al. 2019) proposed an advanced version of DR called Automated Domain Randomization (ADR) to solve a Rubik's cube using a multi-dexterous robotic arm. This proposed method can improve the control policies and vision estimates of the trained model. Additionally, it shows that the model can handle situations that were not seen during the training process. Since the ADR method progressively generates difficult environments, the model is generalized to adapt to changes in the real world rather than domain-specific.

Additionally, domain adaptation (DA) is another transfer learning methodology that allows the model to learn transferable representation from the environment (Wang and Deng 2018). There are two classes of DA: instance-based and feature-based. DA is utilized in (Bousmalis et al. 2018; Yan et al. 2017; Hundt et al. 2020) research to optimize robot grasping. Research has shown that the DA method allows the model to learn a mapping from source to target domain (Tobin et al. 2017). Additionally, the DA method allows us to learn domain invariant features from the source domain (Gupta et al. 2017). Many of the domain adaptation applications include robot grasping (Yan et al. 2017; James et al. 2018) and visual-based navigation (Zhang et al. 2022).

One other method proposed by the research is to create high-fidelity virtual environments to closely match the real-world environment. This involves creating environments with higher photorealism and simulation capabilities. NVIDIA Flex (Kar et al. 2019), Alphabet Soup, MuJoCo and RAWSim-O (Ranaweera and Mahmoud 2021) are some of the high-fidelity GPU-based simulators. Some of these applications require higher computation power for accurate rendering and simulations. Therefore, most researchers in this context used game engines such as Unity and Unreal Engines as they provide highly optimized, photorealistic, real-time, and accurate physics simulations without consuming system resources.

There were shortcomings and limitations in the review papers. Most research articles suggest that utilizing the domain randomization method allows for inducing variability to generalize the model. However, one of the shortcomings of the DR method is that it cannot randomize properties that are not modeled in the environment. Therefore, it requires additional work to create ideal learning environments. OpenAI's implementation of training a multi-dexterous arm to solve a Rubik's cube uses multiple camera systems, an advanced Giiker cube (that consists of multiple sensors), and a consistently lighted environment to make observations. One of the major observations made by OpenAI is, data collected through multiple sources and sensors, requires a high level of calibration and suffers from curse-of-dimensionality. Similarly, this research utilizes external sensors such as a camera and an accelerometer, and 3 axis gyro sensor to track the location of the marble and determine the orientation of the platform. However, based on the RL algorithm used, necessary sensors are selected to minimize the number of dimensions. In addition, this research uses domain randomization to randomize textures, lighting conditions, camera location, and field-of-view to induce variability. Random noise is introduced to the simulation so that it may create additional variability in the simulation to generalize the model further during RL. Two RL learning techniques: Actor-Critic and Deep Q-Learning are utilized to understand the effect of noise and domain randomization on the virtual environment. In this research, the Deep Q-Learning method utilizes raw image data obtained from the virtual camera, whereas the Actor-Critic method utilized marbles position, velocity vector, and relative location to make predictions.

Methodology

In this research, to evaluate the quality Sim2Real policy transfer, two environments were created; a virtual environment to perform the RL learning, and a physical environment to evaluate the quality of transfer and the existence

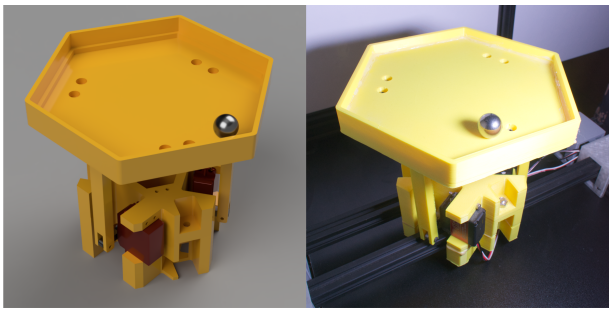

Figure 1: Virtual environment rendered on Godot game engine (left) and the physical environment (right)

of reality-gap. For the testing environment, a 3-DOF Stewart platform was created virtually and physically that are identical as shown on Figure 1. The goal of the learning agent was to balance a simple marble placed on the Steward platform. The research utilized the system architecture outlined in (Ranaweera and Mahmoud 2023) for RL and transfer learning. Both virtual and physical systems consist of identical modules for performing set tasks. The virtual environment contains dynamic simulation, environment, and rendering modules to perform virtual actions, containing a virtual 3-DOF platform, camera, lighting, and rendering engine respectively. The virtual environment was utilized to train the model by performing pre-defined random actions. Once the agent is transferred to the physical system, the goal is to observe the performance and accuracy of the agent while tuning the hyperparameters and environment conditions until optimum performance is achieved. Additionally, both virtual and physical environments were configured to exact environment parameters such as action delay, degrees of freedom, angular limitations, initial camera position, field of view, starting position, and weight.

Virtual Environment

Stewart platform was first drafted and created in Autodesk Inventor to create identical virtual and physical environments. Godot (godotengine.org), an open-source game engine, was used to create the virtual environment. This game engine utilizes the Bullet physics engine which allows it to perform high-fidelity physical simulations and collisions on rigid and soft bodies. For communication, relaying actions, and making observations, a WebSocket server was implemented. Custom scripts were made to change environment settings such as lighting, camera position, textures, and framerate. Additional scripts were created to randomize the marble position, size, weight, and surface friction on collisions. Game engine was configured to use GPU-based rendering to improve the performance and simulation fidelity. To reduce the memory allocation and CPU utilization, the virtual environment was compiled into a standalone executable. Each runnable executable was assigned a unique TCP WebSocket port for communicating when training with multiple instances.

A virtual camera was used to capture the view and was mounted at a similar height relative to the physical camera. The field of view (FOV) was configured programmatically to match the real-world camera. Custom scripts were created to randomize camera height and FOV, and add noise and a gimbal on its axis to induce variability in the observations.

Rigid body and static body attributes were added to 3D meshes of the Stewart Platform to assign dynamics and collision properties for the physics simulations. Custom 3 DOF joints were programmed and assigned to the virtual platform. Servo delay, motion range (minimum range and maxim range), and torque were set as environment variables and were randomized during the training process.

Physical Environment

To assess the learning transferred from the virtual environment, a physical 3-DOF Stewart platform was constructed. To ensure that both virtual and physical environments are identical, the physical platform was 3d printed using the CAD model. A robot cage was constructed to house the physical platform. The robot cage was used to mount the camera, and lighting rig and to stabilize the platform. Dampening was added to reduce the gear backslash and vibrations, affecting the observations.

The system was controlled through a Jetson Nano compute module that allows it to host the trained RL model, capture and process real-time observations, and interface with actuators and sensors to perform actions. There are three actuators for each axis control. For making observations and capturing frames, a generic 1080p camera module was used. OpenCV, an open-source vision library was utilized to interface with the camera module and capture and pre-process image data. The camera was calibrated using a checkerboard calibration method to reduce radial distortion and tangential distortion. Each captured frame was downscaled to 480x480 pixels for calculating the velocity vector and tracking the marble on the platform. The captured frame was further downscaled to 84x84 pixels before using it as input to the Q-Learning methodology.

Reinforcement Learning

Algorithms implemented in previous work (Ranaweera and Mahmoud 2023; 2022) have been utilized and improved upon for this research. Two algorithms were created; Deep Q-learning and Actor-Critic. Each algorithm was developed to use different inputs as observations. For Deep Q-Learning, downsampled raw frames captured through the virtual camera were used as inputs. While for the Actor-Critic algorithm, raw frames were pre-processed to determine the marble's position, velocity vector, and relative location to use as inputs. In the real world, raw images captured through the webcam were processed using the OpenCV library and set as input to the agent. The velocity vector, position of the marble, relative location, and acceleration were also calculated and used as additional input data.

RL architecture implements four main steps to control the environment and learning. These four steps are environment initialization, reset action, step, and rendering. First, the environment was initialized by creating a new virtual environment instance using a random seed. Environment parameters

such as field-of-view, camera position, and lighting conditions were initialized. The reset step allows the environment to be reset when the reward reaches a certain level or if the agent fails to perform the desired action. Rendering steps involve an agent sending a random action or a prediction during the replay after reaching a certain threshold and making an observation by capturing data after the action has been performed. The action space defined for the Stewart platform is to discrete with an action range of [0, 4]. The RL agent learns to perform pitch forward, pitch backward, roll left, and roll right to guide the marble toward the center of the platform. The reward function assigns positive and negative rewards based on the position of the marble. The reward encourages the agent to make an observation and perform an action such that the marble is guided towards the center goal while making observations on the platform. The platform is divided into three sections, where the center is the target goal. If the marble is within the center of the platform +1 reward is awarded. If it is outside the center -0.5 is deducted and if the marble reaches the edge of the platform the environment is reset for a new iteration. Observations are made through the camera, accelerometer, and 3-axis gyroscope.

Domain randomization and noise were induced during the training to increase the generalizability of the RL agents during the Sim2Real transfer. The goal of these techniques is to increase the diversity of the environment and policy learning such that the RL agent is able to adapt and perform well in novel situations or environments that are not encountered during the training. Generalization of the RL agent is a crucial and practical aspect of Sim2Real learning to overcome the reality gap. During domain randomization, the textures of the environment, the position of the marble, lighting conditions, frame rate, marble weight, marble size, friction, camera focus, and other dynamics are changed and randomized.

Domain Randomization and Added Noise

In this research, Domain randomization and induced noise are important to increase the diversity of the training data. In domain randomization, random environments are generated by randomizing pre-defined aspects of the virtual environment; such as the position of the marble, camera, changing lighting conditions, textures, background, and foreground. Figure 2 and Figure 3 demonstrate the applied domain randomizations and examples of the camera and lighting conditions adapted from (Ranaweera and Mahmoud 2023).

Background and foreground randomization was performed by setting a random RGB value between [50, 255]. Additionally, the marble's position, size, and mass were changed within the virtual environment. A spotlight was set to light the platform and simulate the real-world lighting rig. The lighting intensity was increased at a constant value(2.67) and defined to be within the range of [0, 16]. The position and FOV of the spotlight and camera were randomized as shown. Since the physical camera, IMX219 has a range of [20, ∞), the virtual camera was also set to the same FOV. The randomizations on the virtual environment (developed on the Godot game engine) were performed using custom GDScripts.

In addition, custom scripts were developed to perform Gaussian noise on the action space and add noise during the training to further increase the variance. The Action space noise allows for improved exploration of the environment, increases agents' robustness, and allows better convergence while preventing overfitting. As demonstrated by (Plappert et al. 2017), the generalization error will be improved; and thereby, the agent will be able to compensate for the reality gap.

Reinforcement Learning Algorithms

Deep Q-Learning. The deep Q-Learning method uses an off-policy learning method to optimize the Q-function. The goal is to create optimum Q-function by performing actions against the virtual environment. In this research, Q-Learning utilizes a Convolutional Neural Network(CNN) layer to process raw image data. The Image data are captured through the virtual or real camera. Then the image data are downscaled to 84x84 and stacked to represent a sequence. Refer to Figure 8 on (Ranaweera and Mahmoud 2023), which represents the overall model architecture. Based on the image, the Q-Learning agent uses an exploration-exploitation strategy to optimize the Q-function. The agent learns the optimum policy to obtain the maximum reward in the given environment.

Actor-Critic Architecture. The Actor-Critic methodology utilizes two separate models: one for the policy function and the other for the value function. The policy function generates a probability distribution over the set actions space. Action is selected and performed against the environment. The value function is used to determine the expected outcome from the observation. Refer to image On the other hand, Actor-Critic utilizes the marble location, the velocity vector, and the relative position of the marble on the platform. This information is calculated by capturing raw image data from the virtual and physical cameras. Using OpenCV the marble's location is tracked, and by stacking previous frames, the velocity vector is approximated. The velocity vector represents the marble's trajectory on the platform and allows the algorithm to make decisions based on location to guide the marble.

Experiments and Results

Algorithms were trained in a distributed fashion on multiple sessions to increase efficiency and speed up the learning process. The main goal of this research was to identify how each randomization affects the overall policy learning and to evaluate each agent in the physical environment to determine the success rate. The evaluation begins with no randomization and ends with all randomization and noise enabled during the training process. This is to identify the factors that affect learning and to generalize the model further. With domain randomization and induced noise enabled, the agents are exposed to a larger dataset with dynamic and complex environments.

Each training episode was set to 1M frames and when it reached the end, the environment was randomized using a random seed value. The number of episodes per distributed

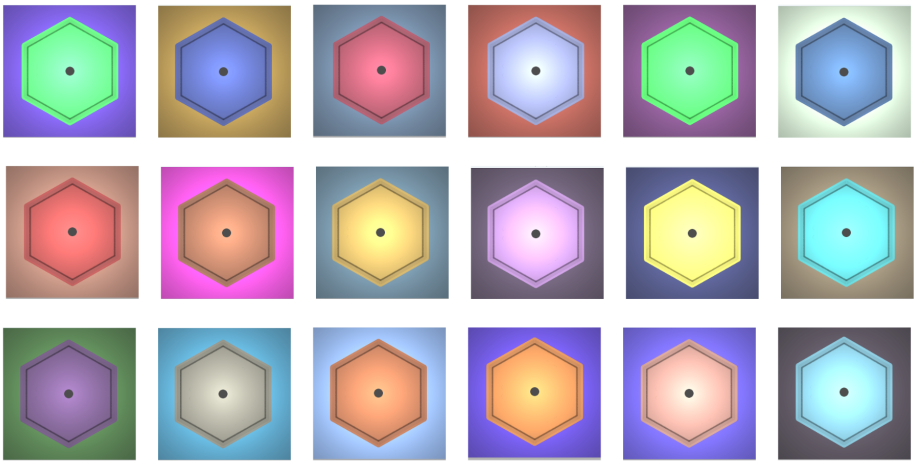

Figure 2: Domain Randomization applied to the virtual environment

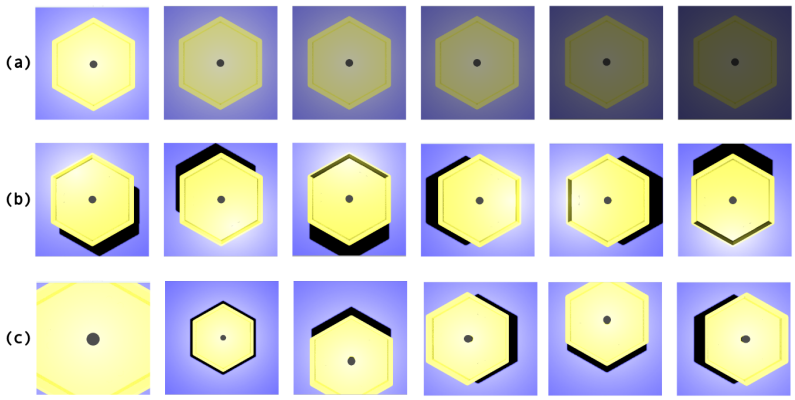

Figure 3: Randomization of (a) Lighting intensity, (b) light position (c) camera position and FOV

learning was set to 1000 during the training. The goal of the agent is to maximize the reward by moving the marble toward the center of the platform. Tensor-board was used for monitoring the training, learning, and replaying.

Once the training had been completed the trained agent was transferred over to Jetson Nano compute module for evaluation in the physical environment. The agent transferred to the physical environment was evaluated for 10 trials to obtain the success rate for each randomization condition outlined in Table 1. The success rate outlines the percentage, in which the platform guides the marble toward the center. The maximum average column denotes the overall average observed in the given randomization. During the training, the RL agent was exposed to increasingly difficult and varying environments through domain randomization and inducing noise. The Table 1 shows the overall performance across each randomization in the virtual and corresponding physical environment.

In our previous study (Ranaweera and Mahmoud 2023), the environment was set to randomize the marble position, camera position and the combined randomization with domain randomization and induced noise. In that experiment, the combined results yield an 89.55% success rate in the virtual and 78.56% accuracy in the physical environment. During this evaluation, optimization to a virtual environment and precise control over the noise levels allows for improvement in the overall performance to an 89.90% success rate in the virtual environment and an 81.45% success rate in the physical environment respectively.

As shown, Actor-Critic outperforms the Q-Learning agents. In both cases, the training environment with domain randomization and induced noise provided higher rates of reward in the physical environment, showing that the model was able to generalize to the physical environment. Q-Learning results show different randomizations and their virtual replay and physical environment results which are drastically different.

Without any randomization, Q-Learning performs the worst in the physical environment. This is due to the inconsistencies between the two environments (domain). Since Q-Learning used the downscaled image data, it seems that physical environmental conditions that were unmodelled affect the final results. On the other hand, since the Actor-Critic algorithm relies on inherited input data, it demon-

Table 1: Experimental results of evaluating the RL agent on virtual and physical environments for Q-Learning (A) and Actor-Critic (B)

(A) Q-Learning	Virtual Environment		Physical Environment	
Environment Configuration	Success	Avg. maximum Reward reached	Success	Avg. maximum reward reached
No randomization	48.34%	2452	20.25%	668
Marble position	82.33%	7563	43.62%	2783
Marble size	56.88%	3246	18.38%	568
Camera fov	67.39%	5892	34.56%	964
Camera position, Camera fov	76.83%	6923	42.34%	234
Lighting position	55.21%	3178	24.78%	857
Lighting intensity	43.53%	3436	17.88%	525
Noise (Gaussian Noise) - on image data	65.89%	5689	56.87%	3235
Noise (Gaussian Noise) - on action space	78.46%	7289	66.39%	5726
With domain randomizaton (all attributes)	86.35%	7854	68.98%	6238
With domain randomizaton and induced noise	**89.90%**	**8361**	**81.45%**	**7325**

(B) Actor-Critic	Virtual Environment		Physical Environment	
Environment Configuration	Success	Avg. maximum Reward reached	Success	Avg. maximum reward reached
No randomization	85.34%	7844	68.74%	6167
Marble position	88.45%	8102	78.67%	6821
Camera position	80.66%	7345	72.56%	6734
Marble size	78.22%	7104	66.67%	5689
Camera fov	81.45%	7389	66.88%	5718
Camera position, Camera fov	83.56%	7655	78.45%	7192
Noise (Gaussain Noise) - on action space	88.98%	8263	82.78%	7658
With domain randomization and induced noise	**96.65%**	**9857**	**86.27%**	**7933**

strates no effect on randomizations such as lighting position and lighting condition. However, the camera position and the field-of-view had an effect on the marble tracking, causing a few failed movements on the physical system. On occasions, due to lighting, specular highlights, and other environmental conditions, the OpenCV could not track the marble precisely which affected the overall performance during the evaluation. However, when overall lighting and camera were positioned correctly, overall performance improved. The Gaussian noise on action space clearly shows an increased success rate in the physical environment. Actor-Critic, the algorithm in this research demonstrates that it requires a high-fidelity environment to perform the successful transfer of learning from Sim2Real. If the goals, objectives, and environmental limitations are clearly defined, the RL algorithms such Actor-Critic are able to learn the policies required.

Conclusion and Future Work

This research was conducted to evaluate methods proposed to bridge the reality gap using RL. Two RL algorithms were implemented: Deep Q-Learning and Actor-Critic. Each algorithm was defined to use different inputs as observations to identify which factors affect the overall ability to learn new policies. To evaluate, a Stewart Platform was designed in a virtual and physical environment. The virtual environment was utilized for RL while the physical environment was used for evaluating. Domain randomization and noise were induced to increase the variability and diversity in the virtual environment.

According to the observations, the Actor-Critic methodology outperforms Q-Learning. Since the actor-Critic relies on positional and velocity data, it was able to learn features and policies that require adaptation to the physical environment. The induced noise allows generalizing of the model, further allowing it to perform on the physical Stewart Platform. Q-Learning, on the other hand, based its prediction on the raw image data acquired through the webcam. However, it performs poorly on the physical platform due to inconsistencies in the system dynamics and environmental lighting conditions.

To further bridge the reality gap, future research will focus on improving Q-Learning by optimizing the DR methodology and identifying factors affecting performance. The fidelity of the multi-agent learning environments should be improved to increase the learning and further generalize the model. Additionally, domain adaptation and transfer learning methodologies will be employed to improve the transfer policies and to further generalize the model. Furthermore, current results will be compared with similar implementations to further bridge the reality gap.

References

Bousmalis, K.; Irpan, A.; Wohlhart, P.; Bai, Y.; Kelcey, M.; Kalakrishnan, M.; Downs, L.; Ibarz, J.; Pastor, P.; Konolige, K.; Levine, S.; and Vanhoucke, V. 2018. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 4243–4250. ISSN: 2577-087X.

Doersch, C., and Zisserman, A. 2019. Sim2real transfer learning for 3D human pose estimation: motion to the rescue. arXiv:1907.02499 [cs].

Gupta, A.; Devin, C.; Liu, Y.; Abbeel, P.; and Levine, S.

2017. Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning. arXiv:1703.02949 [cs].

Hundt, A.; Killeen, B.; Greene, N.; Wu, H.; Kwon, H.; Paxton, C.; and Hager, G. D. 2020. "Good Robot!": Efficient Reinforcement Learning for Multi-Step Visual Tasks with Sim to Real Transfer. *IEEE Robotics and Automation Letters* 5(4):6724–6731. arXiv:1909.11730 [cs].

James, S.; Wohlhart, P.; Kalakrishnan, M.; Kalashnikov, D.; Irpan, A.; Ibarz, J.; Levine, S.; Hadsell, R.; and Bousmalis, K. 2018. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. *CoRR* abs/1812.07252.

Kar, A.; Prakash, A.; Liu, M.-Y.; Cameracci, E.; Yuan, J.; Rusiniak, M.; Acuna, D.; Torralba, A.; and Fidler, S. 2019. Meta-Sim: Learning to Generate Synthetic Datasets. arXiv:1904.11621 [cs].

Lipson, H.; Bongard, J.; Zykov, V.; and Malone, E. 2006. Evolutionary Robotics for Legged Machines: From Simulation to Physical Reality. 11–18.

Lomnitz, M.; Hampel-Arias, Z.; Lopatina, N.; and Mejia, F. A. 2020. A general approach to bridge the reality-gap. arXiv:2009.01865 [cs].

OpenAI; Akkaya, I.; Andrychowicz, M.; Chociej, M.; Litwin, M.; McGrew, B.; Petron, A.; Paino, A.; Plappert, M.; Powell, G.; Ribas, R.; Schneider, J.; Tezak, N.; Tworek, J.; Welinder, P.; Weng, L.; Yuan, Q.; Zaremba, W.; and Zhang, L. 2019. Solving Rubik's Cube with a Robot Hand. arXiv:1910.07113 [cs, stat].

Park, S.; Kim, J.; and Kim, H. J. 2020. Zero-Shot Transfer Learning of a Throwing Task via Domain Randomization. In *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, 1026–1030. ISSN: 2642-3901.

Plappert, M.; Houthooft, R.; Dhariwal, P.; Sidor, S.; Chen, R. Y.; Chen, X.; Asfour, T.; Abbeel, P.; and Andrychowicz, M. 2017. Parameter space noise for exploration. *CoRR* abs/1706.01905.

Ranaweera, M., and Mahmoud, Q. H. 2021. Virtual to real-world transfer learning: A systematic review. *Electronics* 10(12):1491.

Ranaweera, M., and Mahmoud, Q. H. 2022. Bridging Reality Gap Between Virtual and Physical Robot through Domain Randomization and Induced Noise. *Proceedings of the Canadian Conference on Artificial Intelligence*. https://caiac.pubpub.org/pub/kzx3gl4e.

Ranaweera, M., and Mahmoud, Q. H. 2023. Bridging the reality gap between virtual and physical environments through reinforcement learning. *IEEE Access* 11:19914–19927.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. arXiv:1703.06907 [cs].

Wang, M., and Deng, W. 2018. Deep Visual Domain Adaptation: A Survey. *Neurocomputing* 312:135–153. arXiv:1802.03601 [cs].

Yan, M.; Frosio, I.; Tyree, S.; and Kautz, J. 2017. Sim-to-Real Transfer of Accurate Grasping with Eye-In-Hand Observations and Continuous Control.

Zhang, T.; Zhang, K.; Lin, J.; Louie, W.-Y. G.; and Huang, H. 2022. Sim2real learning of obstacle avoidance for robotic manipulators in uncertain environments. *IEEE Robotics and Automation Letters* 7(1):65–72.