applied sciences

MDPI

*Review*

# Sim–Real Mapping of an Image-Based Robot Arm Controller Using Deep Reinforcement Learning

**Minoru Sasaki [1,*], Joseph Muguro [1,2], Fumiya Kitano [3], Waweru Njeri [2] and Kojiro Matsushita [3]**

[1] Intelligent Production Technology Research & Development Center for Aerospace (IPTeCA), Tokai National Higher Education and Research System, Gifu 501-1193, Japan
[2] Center for Robotics and Biomedical Engineering, Dedan Kimathi University of Technology, Nyeri 657-10100, Kenya
[3] Graduate School of Engineering, Gifu University, Gifu 501-1193, Japan
* Correspondence: sasaki@gifu-u.ac.jp; Tel.: +81-90-6462-0957

**Abstract:** Models trained with Deep Reinforcement learning (DRL) have been deployed in various areas of robotics with varying degree of success. To overcome the limitations of data gathering in the real world, DRL training utilizes simulated environments and transfers the learned policy to real-world scenarios, i.e., sim–real transfer. Simulators fail to accurately capture the entire dynamics of the real world, so simulation-trained policies often fail when applied to reality, termed a reality gap (RG). In this paper, we propose a search (mapping) algorithm that takes in real-world observation (images) and maps them to the policy-equivalent images in the simulated environment using a convolution neural network (CNN) model. The two-step training process, DRL policy and a mapping model, overcomes the RG problem with simulated data only. We evaluated the proposed system with a gripping task of a custom-made robot arm in the real world and compared the performance against a conventional DRL sim–real transfer system. The conventional system achieved a 15–57% success rate in gripping operation depending on the position of the target object while the mapping-based sim–real system achieved 100%. The experimental results demonstrated that the proposed DRL with mapping method appropriately corresponded the real world to the simulated environment, confirming that the scheme can achieve high sim–real generalization at significantly low training costs.

**Keywords:** robot arm; reality gap; sim–real; simulated environment; deep reinforcement learning



**Citation:** Sasaki, M.; Muguro, J.; Kitano, F.; Njeri, W.; Matsushita, K. Sim–Real Mapping of an Image-Based Robot Arm Controller Using Deep Reinforcement Learning. *Appl. Sci.* **2022**, *12*, 10277. https://doi.org/10.3390/app122010277

Academic Editor: Alessandro Gasparetto

Received: 5 September 2022
Accepted: 9 October 2022
Published: 12 October 2022



## 1. Introduction

Artificial intelligence is a promising venture that seeks to revolutionize automation and autonomous operations as witnessed in growing interest in the study. Deep reinforcement learning (DRL), one of the many algorithms utilized in artificial intelligence, affords machines the ability to interact with an environment and extract/learn an optimal behavior from the interactions [1]. Learning is arrived at through a trial-and-error process with a corresponding reward for every action taken. This feature of trial-and-error learning is very appealing as training data do not have to be prepared in advance [2].

Reinforcement learning (RL) has been applied in many fields with varying success such as robotics, autonomous driving, and chess games, among others. This paper will focus on the application of DRL in robotics, particularly robot arms. Some of the challenges facing DRL as far as robotics is concerned include but are not limited to the training time/cost and safety of operations during operations [3,4]. In training, due to the trial-and-error nature of RL, performing such explorations in a real-world environment is costly. Besides that, during the exploration phase, the machine/robot is bound to perform unsafe actions before settling on an optimal action plan. As such, the safe and efficient training of robots is indispensable.

Simulators offer a solution to the above issues. In theory, simulators allow the controller to be trained using diverse data, safer procedures, and an added advantage of

scaling up the learning process by parallel trainings, which might be unfeasible in an actual machine. Once the model (policy) is trained in the virtual environment, it is transferred to the actual system, which is referred to as sim–real transfer [5,6].

Sim–real transfer is only effective if the simulator is given a suitably accurate representation of the actual robot and its environment, which in itself is a difficult task. The problem compounds when all dynamics of the environment/robot are captured as this raises the computational costs. As such, sim–real systems consist of a trade-off between the accurate representation of the robot environment and complexity. As a consequence of this, when such models are deployed in the real world, the controller learnt in the simulator degrades, a phenomenon referred to as the reality gap (RG).

In the literature, several approaches are adopted to overcome the RG challenge. Some of these include meta learning [7], domain randomization [8,9], adversarial RL [10,11], and continual and transfer learning [12,13], among others. These approaches can be broadly divided into domain randomization and adaptation strategies.

Instead of meticulously simulating all of the real-world characteristics, the objective of domain randomization is to randomize the simulation environment in order to cover the whole distribution of the real world. A challenge with that is that prior knowledge is needed to determine how much randomization to incorporate in training for a policy robust to sim–real transfer. That is, deciding what randomization to add with a concrete parsimonious explanation for how and why the randomization works [2]. In domain adaptation, the idea is to unify the source domain with the target domain via adaptive strategies ingrained during or after training. In this regard, the adaptation focuses on the mismatch between the two environments and optimizes the transfer. The current paper focuses on adaptation strategies, specifically a mapping protocol from the real world to the virtual world, to deal with the RG that exists in sim–real systems in the gripping motion of a robot arm.

Until now, conventional DRL studies (to distinguish between sim–real studies) have been conducted and applied in robot arm inquiries. Zhang et al. attempted to control a robot using only image information using the RL algorithm and Deep Q-Network [14]. The control target was a 3-DOF robot arm whose goal was to reach a target point with an end-effector. To reduce the difficulty of the task, a 2D simulator was created for the robot to reach the target. In this case, the authors used an input simulator image of $160 \times 320$ pixels that was reduced and gray-scaled to $84 \times 84$ pixels. In this case, the joints of the robot arm were position-controlled and moved $\pm 0.02$ rad per step. For each step, the robot increased, decreased, or did not change the joint angle. The reward setting was determined according to the change in distance between the end-effector and the target position: +1 reward if the distance was closer from the state before the step, $-1$ reward if it was further away, and 0 if it did not change.

Similar research was done by James et al., who they used image-based RL to optimize the grasping motion of a 7-DOF (3-translational and rotational motions of the tip and an extra rotation of the elbow) robot arm in 3D simulations [15]. In the study, they used a 3D simulation to optimize the grasping motion of a cube as the grasping object. The target was for the robot arm to grasp the cube and lift it to a height of 30 cm. It was also confirmed that the task success rate varied depending on the initial angles. However, when the optimal controller obtained in virtual space was implemented on the actual machine, the robot arm moved to the cube but did not close the gripper. These results show that the gripper has limited timing for opening and closing compared to the control of the six joints, making it difficult to effect an accurate closing of the gripper.

Among the sim–real approaches, Liu et al. proposed a real–sim–real transfer learning vision-based robot grasping task [16]. In the study, a task-relevant simulated environment was created utilizing semantic information from a real-world scenario and transformed into a virtual scenario for real-to-sim training using the DRL technique. Other studies that utilize synthetically generated environments are reported in [10,11,17]. The challenge with

these approaches is they need sophisticated processing to re-generate synthetic data that best approximate the real world.

Research in [18] reported vision-based robot operations with an auto-tuned sim–real transfer system utilizing search parameter modeling. The proposal attempts to match simulator system parameters to the real world without updating RL states. In this strategy, a search algorithm predicts whether a sequence of actions and the corresponding system parameters are higher or lower than the real-world values for corresponding observations. The approach reported successful transfers to simulation and reality in environments where other strategies such as domain randomization would fail.

From the review of the literature, the DRL robot control method that feeds back external information obtained through camera images is effective to certain degrees. As such, the present study utilized the visual observation of the environment to perform pick and place robot operations. A sim–real transfer learning was adopted to enhance the training process in the virtual environment. To address the problem of RG, we propose employ image processing techniques as well as a search algorithm that maps reality to simulation. In this way, the current proposed DRL system consists of a search algorithm motivated by [18] but is different in the search modality.

In the proposed approach, an extra layer of a convolutional neural network (CNN) model is added as an attempt to match the observed real-world to a learnt-simulator scene. By mapping the real-world observation to an already learnt scene in the simulator, the policy is able to approximate a true representation of reality, leading to a policy that is likely to succeed in the real world without updating RL states. This study therefore presents a computationally efficient algorithm that combines the DRL system with a CNN mapping model to realize sim–real transfer. In summary, the main contributions of this paper are listed as follows:

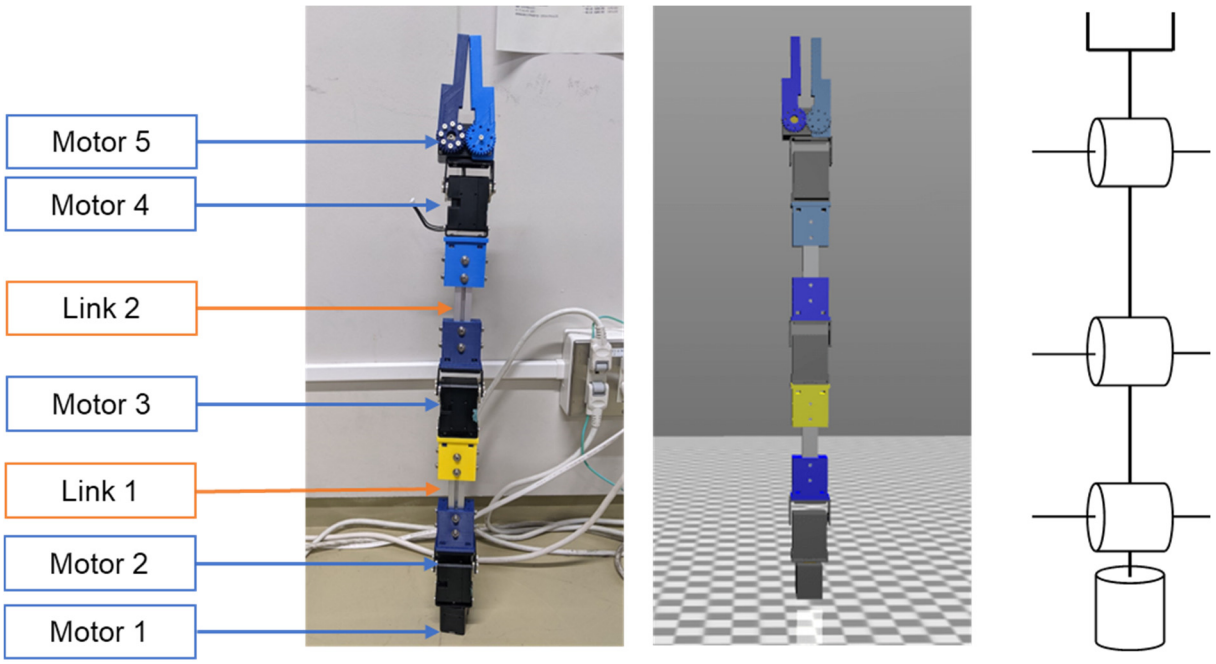
1. We proposed a mapping approach of training a DRL policy to work in the real world for a grasping operation of a robot arm. After training the DRL policy in a virtual environment, we introduced a mapping CNN model that takes in real-world images and maps them to the learnt-simulated environment.
2. We demonstrated the sim–real transfer without any real-world training data.

The rest of the document is organized as follows. Section 2 presents the materials and methods applied in the study. Section 3 describes the performance of the conventional image-based sim–real DRL method in robot gripping operation. Section 4 describes the performance of the proposed sim–real method. Sections 5 and 6 present the discussion and the conclusions drawn from the study, respectively.

## 2. Materials and Methods

### *2.1. Structure of 4-DoF Robot Arm*

The target of the current study is to perform control of a robot arm using DRL strategies. Towards that end, a custom-made 4-DoF robot and a corresponding virtual arm were designed for the study. The actual robot arm has four link joints and one hand joint as shown in Figure 1a. The corresponding virtual model is shown in Figure 1b. Figure 1c shows the functional representation of the robot arm. In the design, the links were made using an aluminum frame. The joints between the links and motor and the gripper were modeled and 3D printed to complete the structural construction. The length and mass of each component are shown in Table 1.




(**a**) (**b**) (**c**)

**Figure 1.** Robot arm configuration. (**a**) Actual robot. (**b**) simulation model. (**c**) 4 DoF representation.

**Table 1.** Robot arm configuration.

| Name | Length [mm] | Wight [g] |
|---|---|---|
| Link1 | 150 | 212 |
| Link2 | 150 | 212 |
| Gripper | 115 | 22 |

A Dynamixel XM540-W150-T smart actuator was used for the motor of the 4-axis robot arm. The XM540-W150-T is capable of position control, speed control, and torque control by general PID control. Table 2 shows the specifications of XM540-W150-T. The control method for each joint is velocity control for the joints and angle control for the gripper.

**Table 2.** The specifications of XM540-W150-T.

| Name | Value |
|---|---|
| Stall torque | 6.9 [N·m] (at 11.1 [V], 4.2 [A])<br>7.3 [N·m] (at 12.0 [V], 4.4 [A])<br>8.9 [N·m] (at 14.8 [V], 5.5 [A]) |
| No-load rotation speed | 50 [rpm] (at 11.1 [V])<br>53 [rpm] (at 12.0 [V])<br>66 [rpm] (at 14.8 [V]) |
| Reduction ratio | 1/152.3 |
| Maximum operating angle | During positioning control: 0 to 360 [deg]<br>(12 bit resolution) |
| Power supply voltage range | 10~14.8 [V] |
| Allowable axial load | 20 [N] |
| Weight | 165 [g] |

### 2.2. Real and Virtual Workspace

The virtual space was constructed using the 3D dynamics engine MujoCo to simulate the motion control of the actual arm. The setup of this study is shown in Figure 2a,b for the virtual and actual space, respectively. In the design of the virtual scene, the color tones, lighting positions, shadows, and reflectance were adjusted to minimize differences from the real space environment.

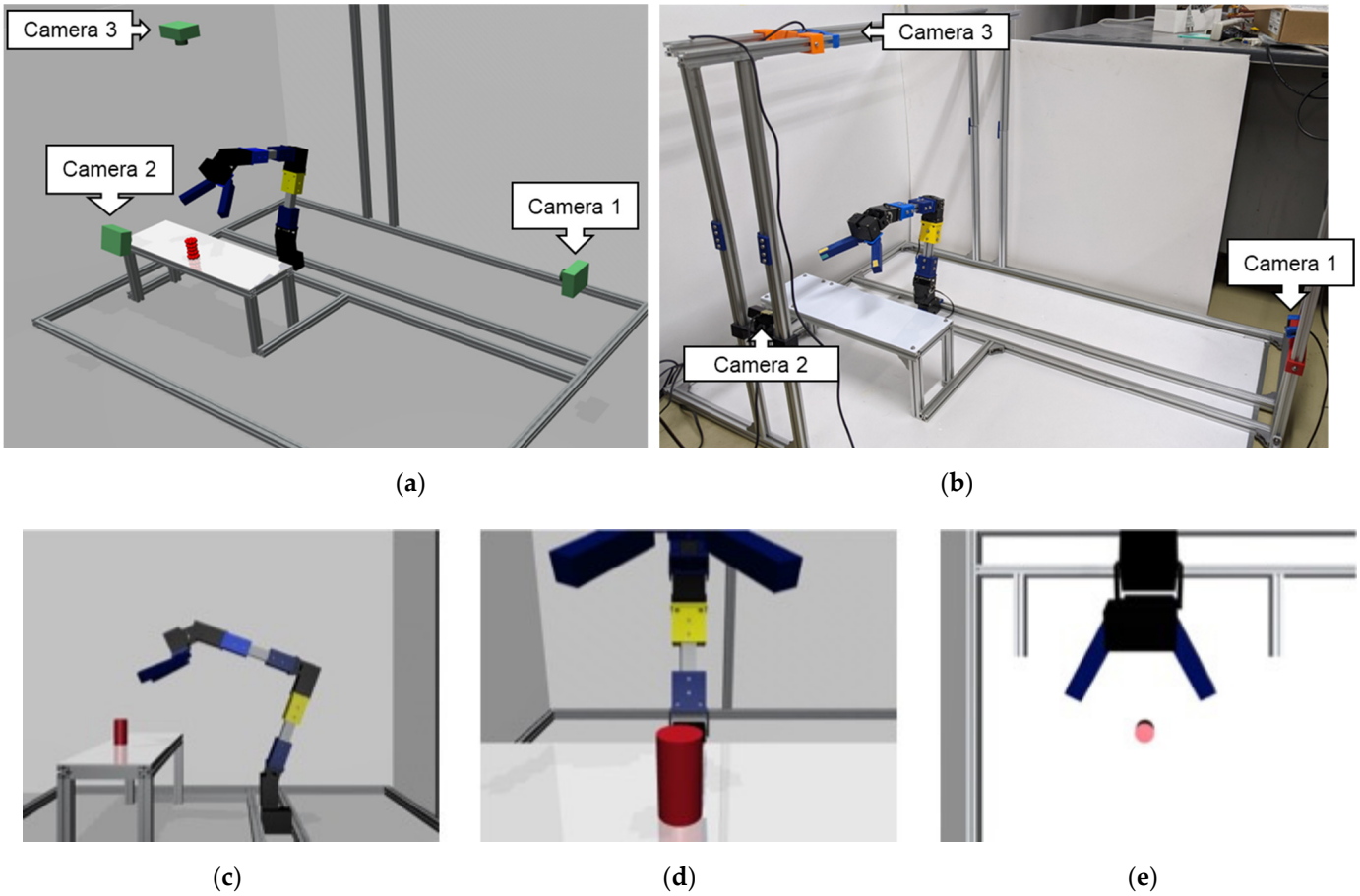
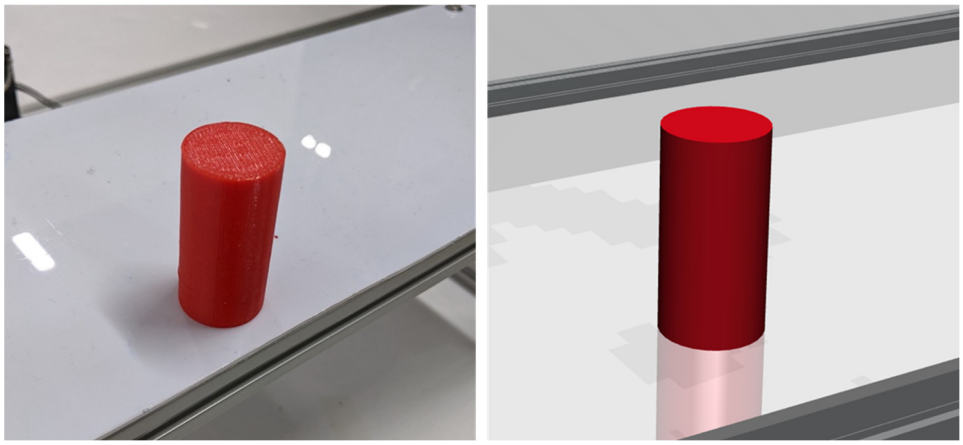



**Figure 2.** The proposed virtual and real workspace and corresponding camera outputs. (**a**) Virtual simulation scene. (**b**) Actual robot setup. (**c**) Camera 1 output. (**d**) Camera 2 output. (**e**) Camera 3 output.

Picking tasks play a very important role in the operation of a robot arm, and it is expected that RL can be used to optimize the operation of picking [19]. In this study, the target is to control a pick-and-place task of a 4-axis robot arm using image inputs. As such, a target (cylinder) is placed on the desk in front of the robot arm, and images are acquired from the x, y, and z directions using three cameras: Figure 2c–e. The robot arm was installed so that the joint state of the robot arm could be seen from camera 1, and the gripper and cylinder could be seen from camera 2 and 3. Each camera location is highlighted in Figure 2 for both the virtual and actual machine.

A cylinder with a height of 50 mm, a diameter of 25 mm, and a mass of 8 g was used as the grasping object as shown in Figure 3. Previous studies implemented 7-DoF with a cube target, thereby making gripper rotation necessary [15]. Contrary to this, the cylinder can be grasped from any angle, hence a 3-rotational axis was not required in the current study. As such, the focus of the study shifted towards efficient learning and not gripping. Gripping operation is handled programmatically and will be described later in this section. A successful grasp is defined as when the robot arm grasps the cylinder and lifts it 100 mm. If the robot arm does not grasp the cylinder or drops the cylinder from the desk, the grasp is considered to be unsuccessful.

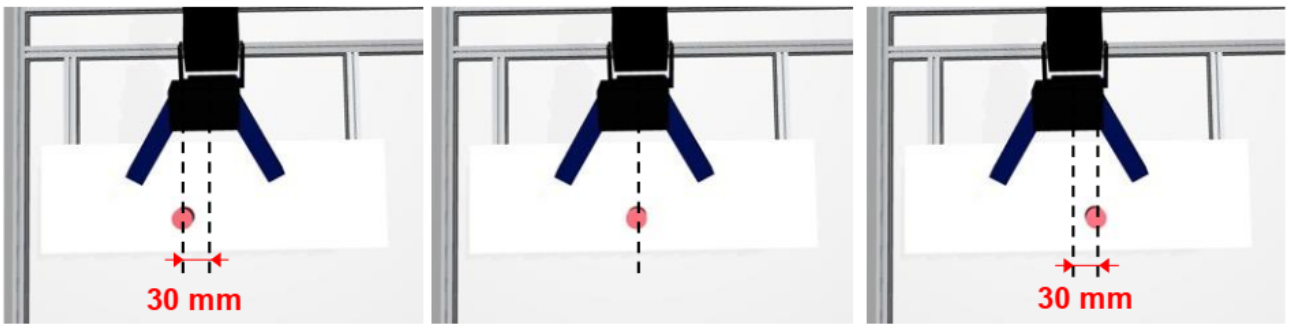


(**a**) (**b**)

**Figure 3.** Grasping object used in the actual machine and simulated environment. (**a**) 3D printed target object. (**b**) Virtual target object.

Previous studies have shown that the task success rate varies depending on the initial position of the robot arm [15]. The task success rate is low when the robot arm is trained at extreme distances from the target position, and the learning process is time-consuming. Therefore, in this study, the initial angles of the robot arm are set as follows, Motor 1 is set to 0°, Motor 2 to 20°, Motor 3 to 90°, and Motor 4 to 45°. The gripper is assumed to be open at 60°.

The placement points of the target were initialized in either of the three positions, left, right, and center as shown in Figure 4. That is, the target was placed ±30 mm in the $x$-axis direction from the center (Figure 4). The reason for this is that the higher the number of placement points, the wider the search space becomes, which increases robustness in learning.




(**a**) (**b**) (**c**)

**Figure 4.** Variation in the initialization position of the grasping target. (**a**) Left. (**b**) Center. (**c**) Right.

### *2.3. Deep Reinforcement Learning System*

Reinforcement learning is a kind of machine learning in which an agent observes the current state in a certain environment and acquires the optimum behavior through trial and error. If the agent takes a certain action stochastically from the current state, the agent will be rewarded by the environment. In 2013, Volodymyr et al. proposed a DRL model using a CNN in the behavioral value function $Q(s, a)$ of RL [20]. Deep Q-Network is a method that combines deep learning with a RL method called Q-learning. The update formula of the action value function $Q(s, a)$ can be expressed as shown in (1), assuming the state $s_t$, action $a_t$, and reward $r_t$ at time $t$. The update process involves selecting the action with the highest value in the state $s_{t+1}$ and updating it with the obtained immediate reward $r_{t+1}$.

When renewing, the value is discounted at the discount rate $\gamma$ because the future reward is uncertain.

$$Q(s_t,\ a_t) = Q(s_t,\ a_t) + \alpha\left(r_{t+1} + \gamma \max_{a_{t+1} \in A(s_{t+1})} Q(s_{t+1}, a_{t+1}) - Q(s_t,\ a_t)\right) \quad (1)$$

From the above, when the number of states becomes enormous, it is difficult to calculate the action value function $Q(s,\ a)$. Hasselt et al. proposed a method to reduce the overestimation of the behavioral value by using two types of networks, a network to be learned and a network for calculating value: Double Deep Q-Network [21]. The update formulae and states are shown in Equations (2) and (3). The Double Deep Q-Network is used as the algorithm in the proposed system.

$$a_{main} = \max_{a_{t+1} \in A(s_{t+1})} Q(s_{t+1}, a_{t+1}) \quad (2)$$

$$Q(s_t,\ a_t) = Q(s_t,\ a_t) + \alpha\left(r_{t+1} + \gamma \max_{a_{t+1} \in A(s_{t+1})} Q\prime(s_{t+1}, a_{t+1}) - Q(s_t,\ a_t)\right) \quad (3)$$

Figure 5 shows the proposed system that takes in the camera images of the real or virtual space and feeds it to deep learning CNN for inference. The image of 480 × 640 pixels acquired from three cameras is preprocessed to reduce the dimensions to 64 × 64 pixels. In visual-based ML, the computational load as well as the network size increase with image dimension. Dimensionality reduction is one of the approaches adopted in RL literature to reduce model complexity while maintaining performance [22,23]. Next, an image from either virtual space or a real camera is passed to the training/inference deep reinforcement learning model. The model outputs discrete actions affecting (increasing/decreasing) the motor values of individual joints. The robot arm is connected to the PC via serial communication.

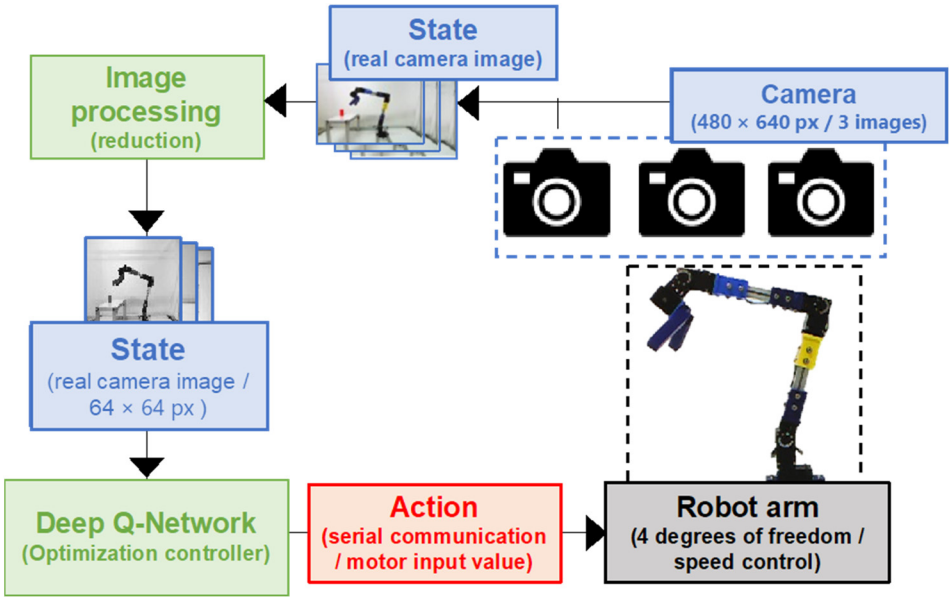



**Figure 5.** Proposed deep learning control scheme.

Deep Q-Network uses a CNN composed of three convolutional layers/pooling layers and two fully connected layers. In this case, the CNN was set up with a 5 × 5 kernel size with single stride. The kernel size of the pooling layer was 2 × 2, and Max pooling was used. The activation function of all layers was adapted to the ReLU function, and Adam was used as the optimization function. The architecture is shown in Figure 6. The input

image from each of the three cameras was 64 × 64 pixels, and the output was the number of actions (10 actions). The Huber loss function was applied to the error function. The output drives the corresponding motor as per the action set described below.

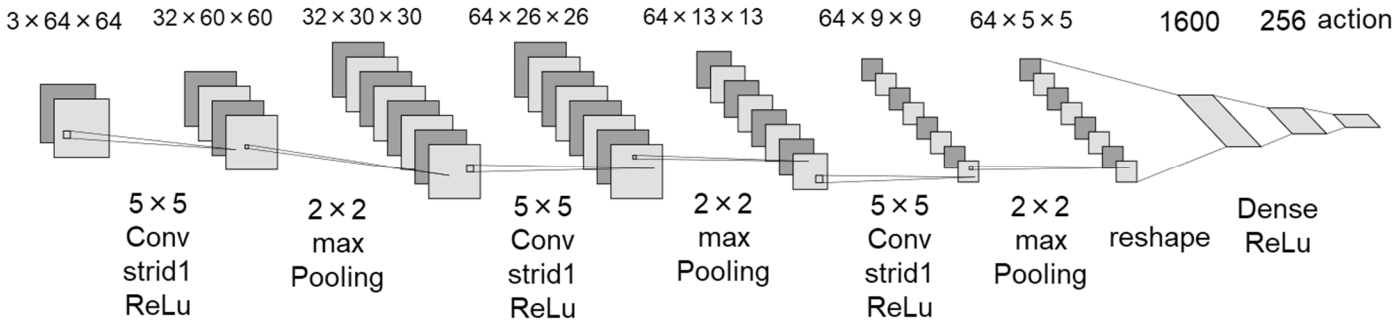



**Figure 6.** The architecture of the Q-Network.

In this experiment, the search space for the RL model was set as a discrete space with 10 action sets as shown in Table 3. From this, three distinct actions can be set for each joint: clockwise: R, anti-clockwise: L, and stop: S. The fourth action is reserved for the gripper as shown in action 10, "close: C". When either R or L action is invoked for any motor, the motor value is altered, corresponding to the input until a new input is supplied. The movement speed was set to 22.9 rpm.

**Table 3.** Combination of actions.

| | Motor 1 | Motor 2 | Motor 3 | Motor 4 | Motor 5 |
|---|---|---|---|---|---|
| Action 1 | S | S | S | S | S |
| Action 2 | R | S | S | S | S |
| Action 3 | L | S | S | S | S |
| Action 4 | S | R | S | S | S |
| Action 5 | S | L | S | S | S |
| Action 6 | S | S | R | S | S |
| Action 7 | S | S | L | S | S |
| Action 8 | S | S | S | R | S |
| Action 9 | S | S | S | L | S |
| Action 10 | S | S | S | S | C |

In the study, default training parameters were adopted with the alteration of the step size during the initial tuning process. The final parameter settings for the RL Deep Q-Network are shown in Table 4. The learning rate was set to 0.0001 and the maximum number of steps to 300. The sampling period was set to 0.15 [s], considering the performance of the PC used. The discount rate was set to 0.95, so that the value of actions that could complete the task in a short time would be high.

**Table 4.** Learning parameter settings.

| Parameter | Value |
|---|---|
| Learning rate | 0.0001 |
| Maximum number of steps | 300 |
| Sampling cycle | 0.15 [s] |
| Discount rate | 0.95 |
| Replay memory size | $1 \times 10^6$ |
| Start epsilon | 1 |
| Final epsilon | 0.1 |
| Epsilon attenuation coefficient | $6 \times 10^7$ |
| Model update interval in steps | 10 |
| Target model update interval in steps | 1000 |

The reward settings for completed operations are described in Table 5. The reward setting was given so that it was inversely proportional to the distance between the robot hand and the cylinder, and the closer to the cylinder, the higher the reward. In addition, when the distance between the robot hand and the cylinder is within 10 mm, the value of the approaching motion is increased by setting it to +1. If the grip was successful, +1000 was given.

**Table 5.** Reward setting.

| State | Reward |
|---|---|
| Successful grip | 1000 |
| The distance between the robot hand and the cylinder is within 1 mm | 1 |
| When not grasping | $1/(1 + distant)$ |

In the gripping operation, the trained DRL model performs operations to approach the cylinder. When the target is within 10 mm distance, the gripping operation is initiated programmatically, i.e., closing the gripper. This operation takes 50 steps to close the gripper and 20 steps to raise the picked object.

## 3. Sim–Real Model Performance

In this section, we test the performance of the sim–real model that was trained on a simulated scene and review how it performs in the actual world using image processing as a mediator between real and virtual workspaces. In this case, a conventional DRL model is utilized with low-dimensional images, i.e., gray-scaling, binarization, etc., in an actual robot. First, the grasping motion of a four-axis robot arm is optimized in virtual space by adapting DRL and then, the learned model is implemented on the actual machine.

### *3.1. Image Processing for Environment Matching*

The objective of the research was to obtain image-based robot control from the simulated environment. The challenge of such an approach is how the simulated environment mirrors the actual workspace. To address this, structural design-matching was considered as explained in Section 2.2, but this does not guarantee true uniformity. To this end, we adopted image preprocessing to eliminate chromatic information and sensor noise in a bid to increase mirroring as well as achieve dimension reduction in the inference process. Image processing has been used in machine learning and fields, including sim–real transfer, as a means to enhance training [24–27].

In the study, six types of image processing were adopted: grayscale, binarization, 4-level grayscale, 10-level grayscale, 4-level grayscale with a median filter, and 10-level grayscale with a median filter. The kernel size of the median filter was set to $3 \times 3$. Sample scenes are shown in Figure 7a–e to showcase the processing strategies utilized. From the different operations, the background details and target object are impacted differently.

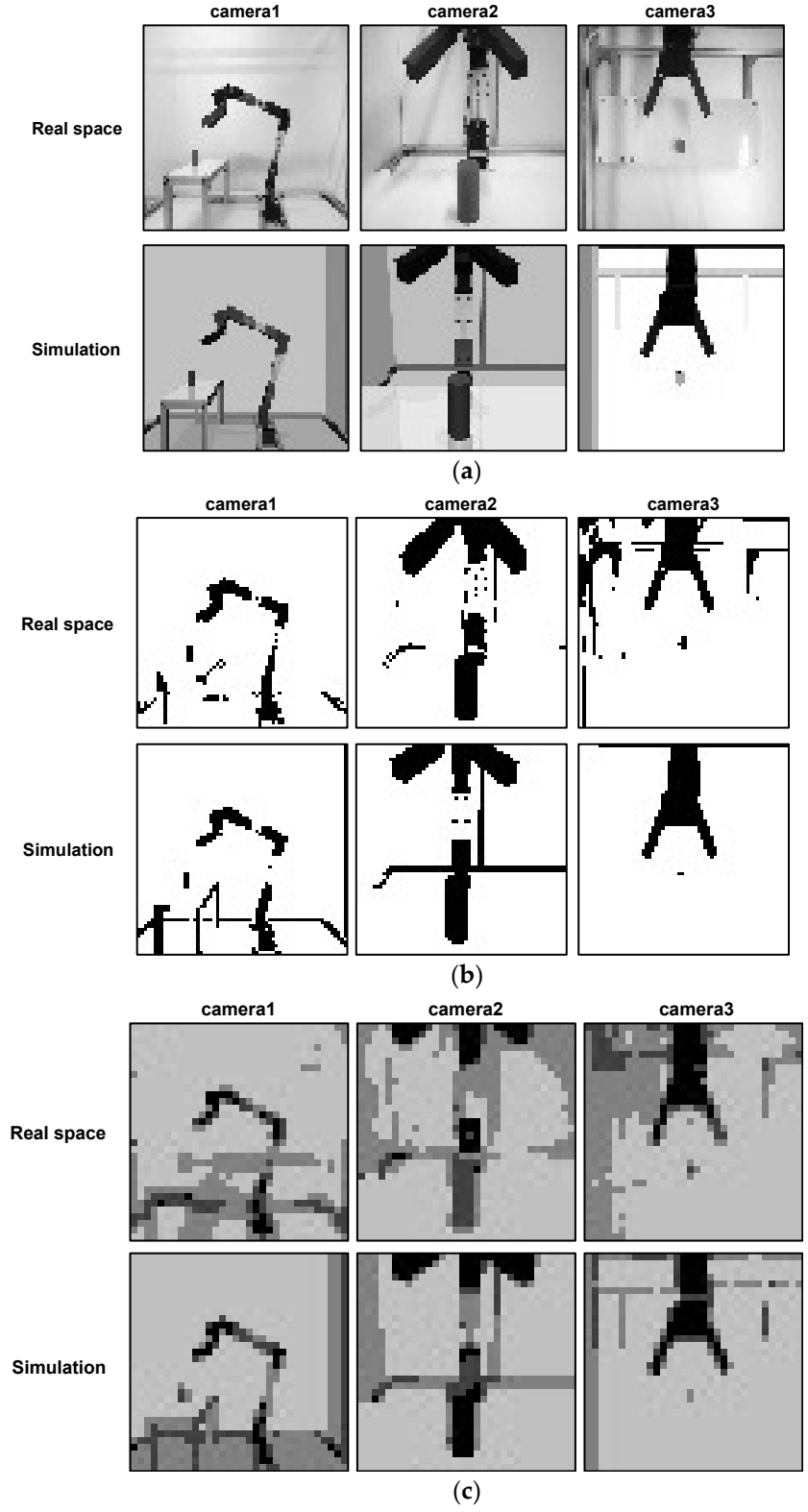



**Figure 7.** *Cont.*

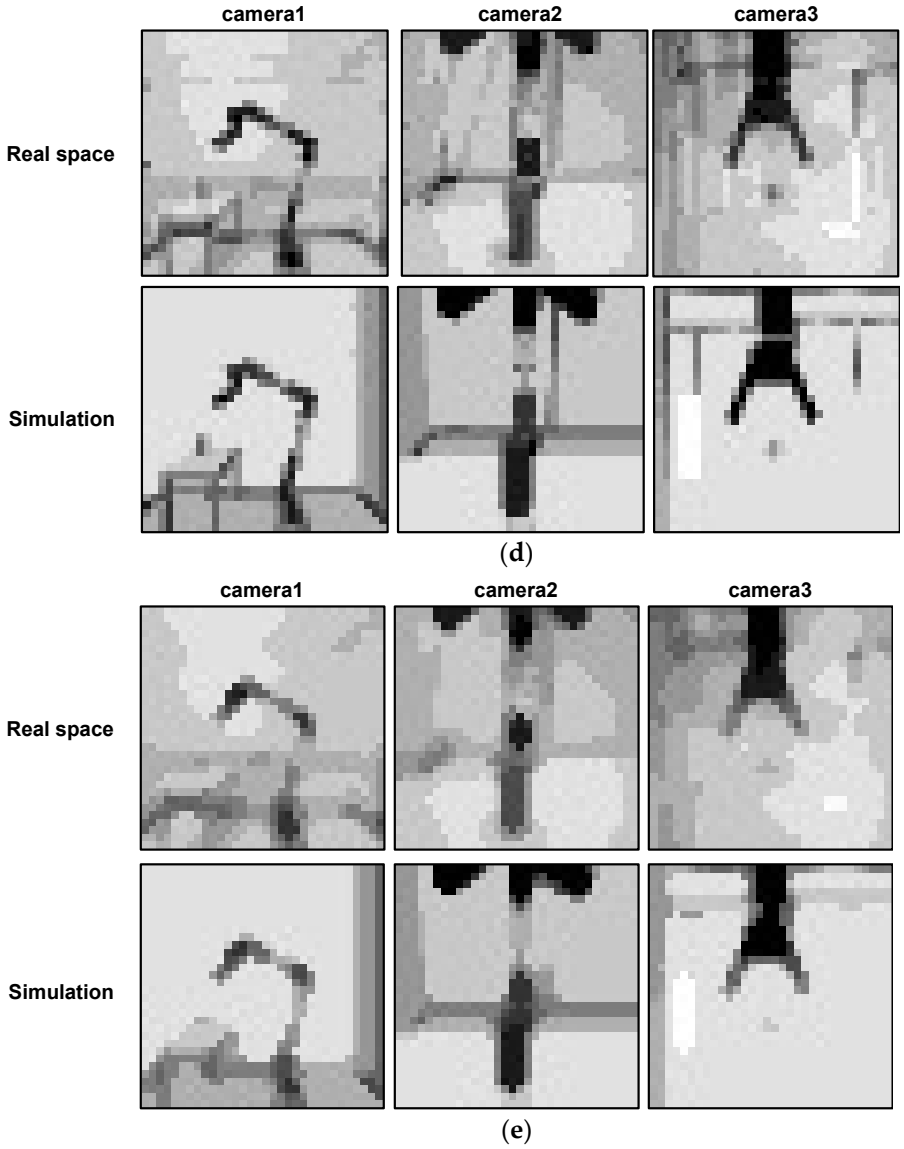
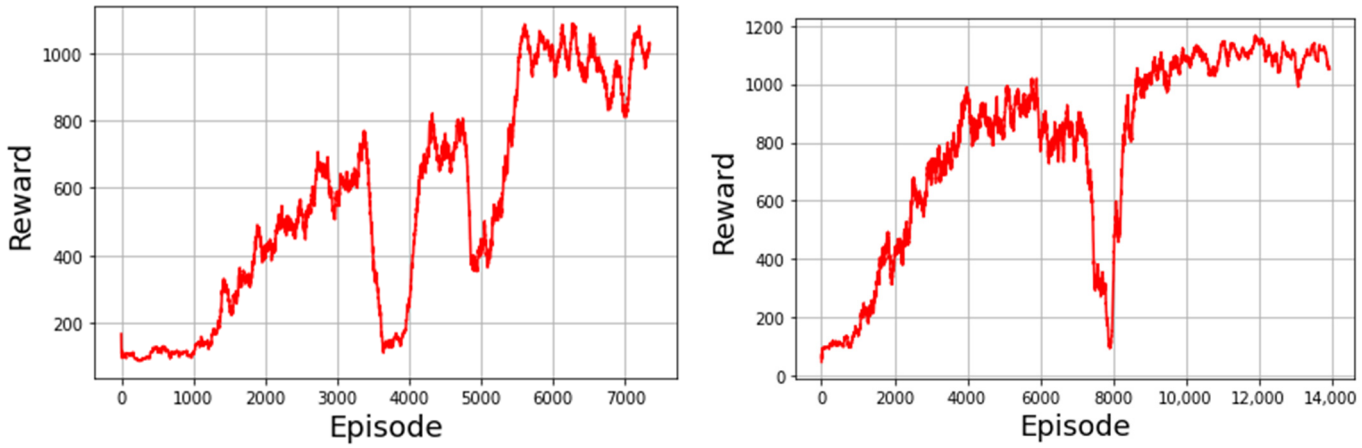



**Figure 7.** Sample scenes of different image processing operations adopted. (**a**) gray scale operation. (**b**) binarization operation. (**c**) 4-gradiation grayscale. (**d**) 10-gradiation grayscale. (**e**) 10-gradiation grayscale with median filter.

### *3.2. Model Gripping Operation*

Figure 8 shows the learning curve of target gripping in the study. The figure shows the moving average of the reward in grayscale and the binarization process. In all of the schemes used, the reward increases as the number of episodes increases as expected, indicating the successful training of the model. It is worth noting that all of the models obtained a satisfactory training curve within 10,000 to 15,000 episodes. From the figure, there was an observed drop in rewards as training progressed. During the dips, we observed situations where the robot moved to a specific place regardless of whether the target was placed there or not, i.e., over-adapted to a specific gripping location. This was thought to be a consequence of incomplete hyperparameter tuning, i.e., the size of the experience replay memory and update methods. When the update interval is short and the memory size is small, the model may be overfit to the most recent action history. From the above, it is necessary to investigate hyperparameter tuning towards the target network update method and loss function, among others. In the current use case, the overall training time was not so critical since the model converged within an acceptable episode length.




(**a**) (**b**)

**Figure 8.** Learning curve in training using the sample image processing technique. (**a**) Grayscale operation. (**b**) Binarization operation.

The cylinders (target objects) were placed at three points (left, center, and right), and the results of five verifications at each point are shown Table 6. In all cases, the robot succeeded in grasping the object on the desk in every instance, confirming that the policy had been sufficiently learnt.

**Table 6.** Results of grasping in the simulation environment.

| Image Processing | Left [%] | Center [%] | Right [%] | Total [%] |
|---|---|---|---|---|
| Grayscale | 100 | 100 | 100 | 100 |
| Binarization | 100 | 100 | 100 | 100 |
| 10-gradation gray scale | 100 | 100 | 100 | 100 |
| 4-gradation gray scale | 100 | 100 | 100 | 100 |
| 10-gradation gray scale w/median filter | 100 | 100 | 100 | 100 |
| 4-gradation gray scale w/median filter | 100 | 100 | 100 | 100 |

### *3.3. Gripping by the Actual Robot*

In the actual machine, the external cameras attached to the system are utilized to supply the images of the setup and feedback. Thus, the motion optimized in the virtual space is reproduced in the real space. Table 7 shows the results of verification (10 times) for each arrangement of the target cylinder. It can be seen that in all cases, the gripping success rate was lower than the result derived in the virtual space (Table 6). In particular, in the case of grayscale and 10-gradation grayscale median filters, it was never possible to grasp the object. A possible reason for the failure of grasping in real space is that the difference between the virtual and real-space images could not be eliminated by reducing the image dimensionality or image processing alone.

**Table 7.** Experimental results in real space.

| Image Processing | Left [%] | Center [%] | Right [%] | Total [%] |
|---|---|---|---|---|
| Grayscale | 0 | 0 | 0 | 0 |
| Binarization | 50 | 100 | 60 | 70 |
| 10-gradation gray scale | 40 | 40 | 70 | 50 |
| 4-gradation gray scale | 0 | 100 | 0 | 33 |
| 10-gradation gray scale w/median filter | 0 | 0 | 0 | 0 |
| 4-gradation gray scale w/median filter | 0 | 100 | 0 | 33 |

From the table, the success rate for each cylinder position was 15% for the left, 57% for the center, and 43% for the right. When placed in the center, only Motor 2 had to be operated to approach, but when placed on the left and right, multiple motors had to be operated, which increased the difficulty of the task. From the above, the results of attempts to implement a system on a real machine using the conventional method show that the system could not behave in the same way as in a virtual space. It can be said that the reality gap cannot be erased simply by image processing strategies alone.

## 4. Real–Sim Mapping

As described in the preceding section, sim–real transfer experiences the RG challenge when deployed in the real environment. The study adopted a mapping strategy that searches for patterns in real space that best correspond to the virtual space. The premise in this approach is that the simulation DRL has been optimally trained to perform successful operations; the only hurdle is the disconnect between the two environments. The role of mapping would therefore be a go-between of observations and learnt experience. In this case, a mapping model is supplied before the DRL model to bridge the gap.

### *4.1. Real–Sim Model*

The proposed real-to-sim mapping model is shown in Figure 9 where the real camera inputs are passed through a CNN model before feeding them to the DRL model. From this, the robot utilizes the already trained model without the need for further adaptations.

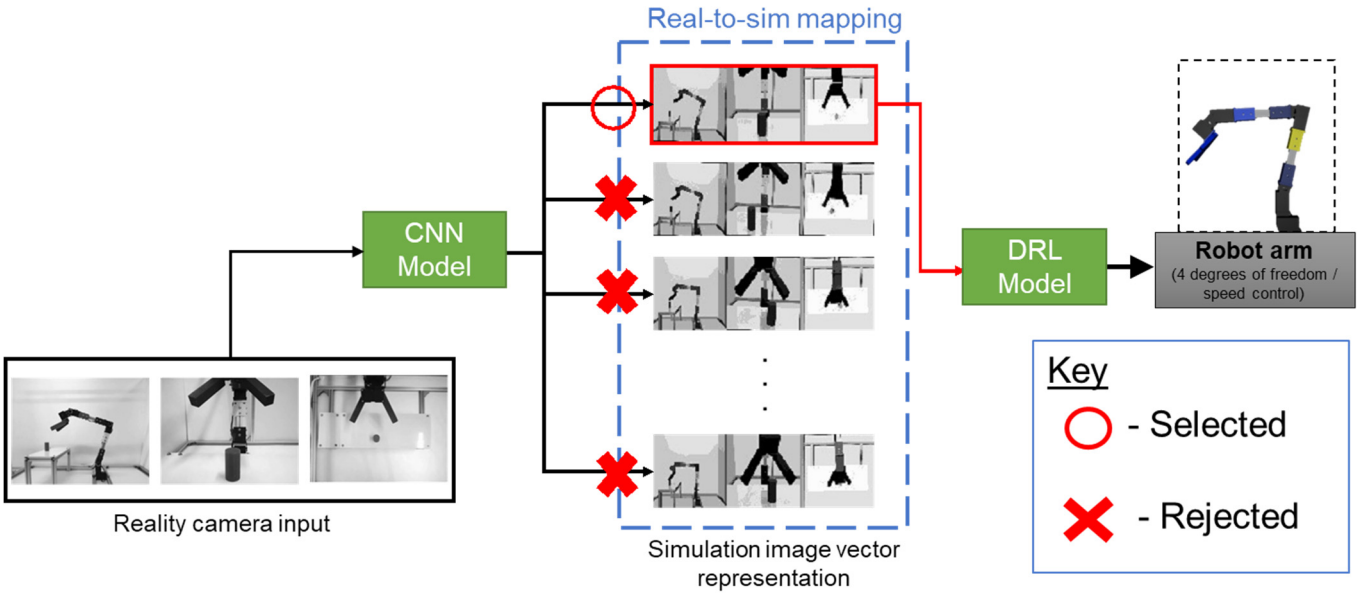



**Figure 9.** Proposed real–sim mapping model showing the selection of the best corresponding virtual image set from the real-world camera input.

To achieve the mapping, the key frames of a successfully executed trial run of DRL robot control in a simulated environment are recorded and saved as a reference database. In this case, 60–70 images (frames) of a successful operation are saved in varying stages of the robot motion as shown in Table 8. After that, random images in subsequent successful trial runs are recorded to create a CNN training dataset as described in Table 9. The CNN model performs a classification of the current input to the nearest approximation of the simulation database.

**Table 8.** Key frames (labels) per processing strategy.

| Image Processing | Key Frames (Labels) |
|---|---|
| Grayscale | 65 |
| Binarization | 64 |
| 10-gradation gray scale | 62 |
| 4-gradation gray scale | 59 |
| 10-gradation gray scale w/median filter | 63 |
| 4-gradation gray scale w/median filter | 67 |

**Table 9.** Deep learning CNN dataset.

| Image Processing | Training Data | Validation Data | Total |
|---|---|---|---|
| Grayscale | 7020 | 1755 | 8775 |
| Binarization | 6912 | 1728 | 8640 |
| 10-gradation gray scale | 6696 | 1674 | 8370 |
| 4-gradation gray scale | 6404 | 1601 | 8005 |
| 10-gradation gray scale w/median filter | 6804 | 1701 | 8505 |
| 4-gradation gray scale w/median filter | 7128 | 1782 | 8910 |

The dataset in use is shown in Table 9. For training, the database was split into 80:20 for training and testing (validation). To supplement the training, data augmentation was performed on the dataset by adding variations (noise) in images that are expected in real-world robot operations, e.g., image rotations, pixel shifting, and blurs, amongst others.

The architecture of the CNN model is shown in Figure 10 consisting of four convolutional and pooling layers, followed by one fully connected layer. The convolutional layer has a kernel size of 3 × 3 and a stride of 1. The pooling layer has a kernel size of 2 × 2 and uses Max pooling. Dropout is applied to the all-coupling layer. The output layer adapts the SoftMax function, the other layers adapt the ReLU function, and Adam is used as the optimization function. Each input image was concatenated to 64 × 192 pixels from image processing. The loss function is the cross-entropy error.

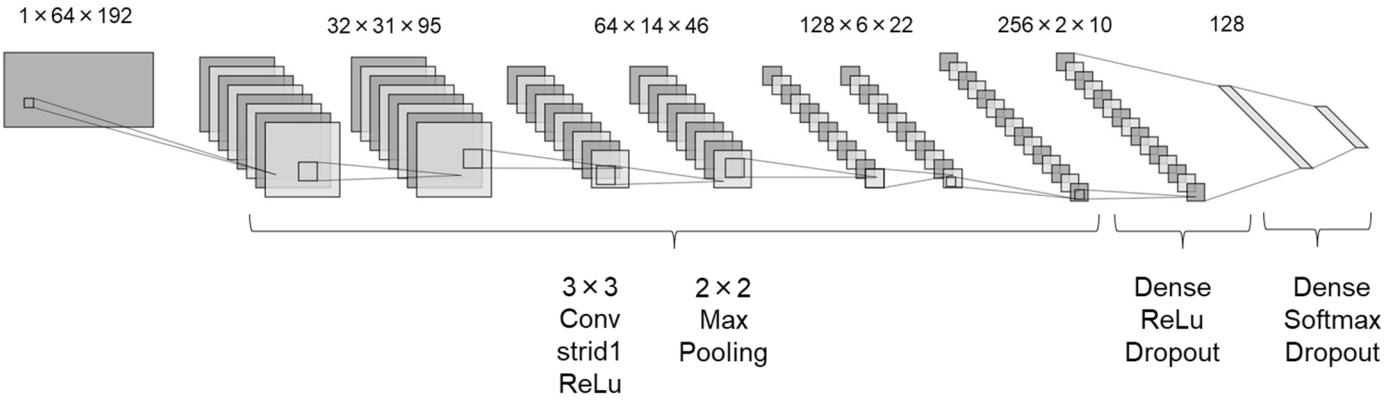
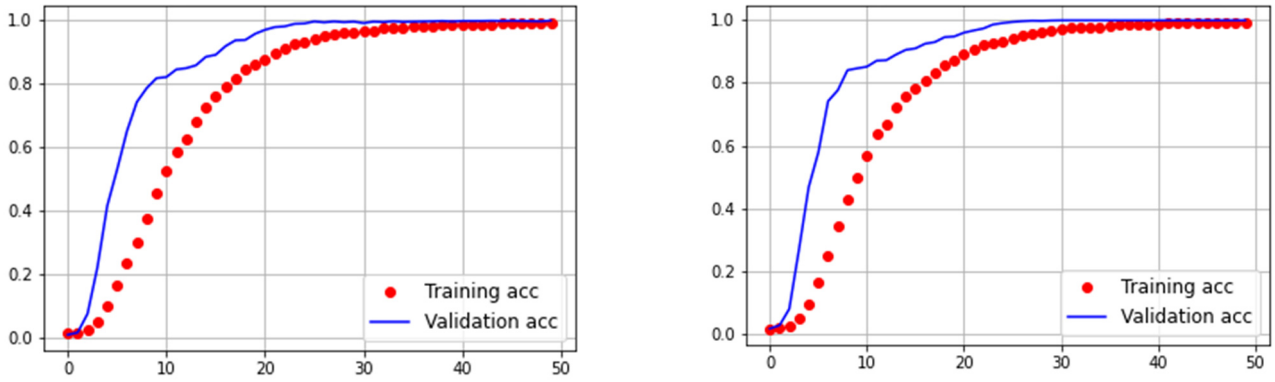



**Figure 10.** Architecture of the CNN model in use.

A trained model was created for each of the image processing strategies supplied in the previous section. The accuracy of the mapping was successful for all of the dimension reduction strategies adopted in the study. Figure 11 shows the learning curve of the grayscale and binarization strategies, as a representative sample of the training process. The accuracy converges to approximately 100% around 40 epochs for all image processing, indicating that a model capable of mapping has been successfully obtained.




(**a**) (**b**)

**Figure 11.** Sample training and validation curves of the CNN mapping operation. (**a**) Grayscale operation. (**b**) Binarization operation.

### *4.2. Gripping by the Actual Robot*

Next, we validated the proposed system using the trained deep-learning CNN and RL model discussed in the previous section. Table 10 shows the grasping success rate for each image processing. The experiments were conducted five times for the left, center, and right cylinder positions, and it can be seen that the grasping was successful in all cases and the grasping success rate was improved compared to the conventional method.

**Table 10.** Experimental results in real space with the proposed scheme.

| Image Processing | Left [%] | Center [%] | Right [%] | Total [%] |
|---|---|---|---|---|
| Grayscale | 100 | 100 | 100 | 100 |
| Binarization | 100 | 100 | 100 | 100 |
| 10-gradation gray scale | 100 | 100 | 100 | 100 |
| 4-gradation gray scale | 100 | 100 | 100 | 100 |
| 10-gradation gray scale w/median filter | 100 | 100 | 100 | 100 |
| 4-gradation gray scale w/median filter | 100 | 100 | 100 | 100 |

Since the grasping was successful in all cases, it can be said that the deep learning CNN was able to form a correspondence between the real and virtual space images.

## 5. Discussion

The paper proposed a sim–real and a corresponding real–sim model to transfer the learnt controller of a robotic arm from the simulation environment to real space. A custom-made 4DoF robot arm was designed for testing the proposed system. The simulation was conducted in the MuJoCo virtual environment, with careful design consideration to mirror the physical setup in an attempt to reduce the reality gap.

In the study, image processing and mapping techniques were applied to reduce the RG between the simulated and actual environment. For image processing, dimensionality reduction and filtering techniques were investigated with the end goal of bridging the reality gap. The advantage of using grayscaled images as opposed to raw images was demonstrated to reduce the computational cost and enhance the accuracy of the network [25].

In the sim–real model, the robot was trained and effectively attained gripping operations successfully for all of the image processing techniques. This is as reported in Table 6 with an average success rate of 100% for all target positions. However, upon deploying the model to the actual robot, the average successful grasps dropped to 15% for the left, 57% for the center, and 43% for the right target positions. This paints a clear picture of the severity of RG, where both the environment and the target objects ought to be similar for

the accurate transfer of knowledge. Approaches of a synthetically generated simulation environment based off the real environment, akin to real–sim–real systems and GAN-related studies, can offer a potential solution in this regard [11,16,17,26]. The challenge would be in complexity in the training environment. As noted by [6], these approaches are inherently data-intensive, thus limiting the practical applicability of such methods. The training time of the approaches is in the range of 100,000 episodes with 1,000,000+ episodes being typical even with high-end computing hardware as reported in [17,26].

The alternative approach considered in this paper is that of mapping the real environment to approximate the simulated environment. In this approach, similar to other reported methods, a search algorithm is introduced to mediate or adapt the real environment to the simulated environment [9,18,28,29]. In the study, a CNN layer is introduced with simulation data to map the real environment to a potentially similar environment in an already learnt simulation environment. With this adjustment, the DRL model attained a 100% grasping success rate in an actual robot arm. Since the grasping was successful in all cases, it can be said that the deep learning CNN was able to form a correspondence between the real and virtual space images. From this, real–sim mapping was confirmed to be more suitable than enhancing the similarity of the two environments.

The advantage of the proposed scheme is the integration with a trained simulation model without requiring any additional RL updates. Further, the image processing method adopted reduces the amount of data to be processed, allowing for a quick and efficient learning process with minimal data requirement as reported in the learning curves (Figures 8 and 11). In the proposal, the mapping model is introduced after a trained policy is attained. This means that it is possible to integrate this with other RG mitigation mechanisms such as domain randomization. The mapping can thus be looked at as a re-sampling of the entire observation to what is critical to the success of the task. From this, the generation of the reference database is critical in the training process. How to choose the keyframes to best capture the entire dynamics of the environment can impact the search algorithm. In the current implementation, we used time-based capturing of keyframes, where a keyframe is saved after every 20–50 updates. More refinement of the algorithm should be done targeting this and other emerging issues.

The current proposal has several limitations, one of which is similar to many other image-based systems, i.e., the approach only addresses the visual gap, and not any physics-based simulation-to-real differences. In the experiment, the motor speed in the gravity direction in the actual machine was set as 11.45 rpm, which is different from what was used in the simulation. When the motor speed was set to 22.9 rpm, to match the virtual environment, the robot moved at a higher speed than expected, which in turn affected the prediction accuracy of the deep learning CNN and altered the system grasping operation. In this case, the effects of gravity and other physics-induced dynamics such as the grasping force, friction, etc., are unaccounted for by the proposed system.

Another limitation is the test environments used, as the evaluation in this study only confirmed the efficacy of the proposal in pick-and-place scenarios. Further research will be conducted to investigate the operations with a wider range of tasks and dynamic environmental settings to further validate the proposal. In the next step, a comparison or integration of other RG adaptation methods will be conducted to prove the efficacy of the proposal.

## 6. Conclusions

In this study, we sought to optimize the gripping operation of an image-based 4DoF robot controller using deep reinforcement learning. We performed the successful training of the gripping operation in the MuJoCo simulation environment and deployed the model in an actual robot arm. The sim–real system achieved a 15–57% success rate in gripping operation depending on the initialization position of the target object. The failure in the gripping process was attributed to the reality gap that exists in simulation-to-reality transfers.

As a remedy to the RG experienced, we adapted the model with a search algorithm that takes in real-world images and maps them to a learnt model in the simulated environment, i.e., sim–real mapping. The strategy achieved a 100% success rate in real robot deployment. The sim–real mapping works best with visual information but fails to account for physics-based reality. Further inquiries are needed to test the performance of the system for wider robot tasks and the complexity of the environment in the future.

**Author Contributions:** Conceptualization, M.S. and F.K.; methodology, F.K. and J.M.; software, F.K.; validation, F.K., M.S. and K.M.; formal analysis, F.K. and J.M.; investigation, F.K.; resources, M.S. and K.M.; data curation, F.K.; writing—original draft preparation, J.M.; writing—review and editing, W.N.; visualization, J.M.; supervision, M.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** This work is partially supported by Grants-in-aid for the Promotion of Regional Industry-University-Government Collaboration from the Cabinet Office, Japan.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Liu, R.; Nageotte, F.; Zanne, P.; de Mathelin, M.; Dresp, B. Deep Reinforcement Learning for the Control of Robotic Manipulation: A Focussed Mini-Review. *Robotics* **2021**, *10*, 22. [CrossRef]
2. Zhao, W.; Queralta, J.P.; Westerlund, T. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, ACT, Australia, 1–4 December 2020; pp. 737–744. [CrossRef]
3. Munikoti, S.; Agarwal, D.; Das, L.; Halappanavar, M.; Natarajan, B. Challenges and Opportunities in Deep Reinforcement Learning with Graph Neural Networks: A Comprehensive review of Algorithms and Applications. *arXiv* **2022**, arXiv:2206.07922.
4. Dulac-Arnold, G.; Mankowitz, D.J.; Hester, T. Challenges of Real-World Reinforcement Learning. *arXiv* **2019**, arXiv:1904.12901.
5. Breyer, M.; Furrer, F.; Novkovic, T.; Siegwart, R.Y.; Nieto, J.I. Flexible Robotic Grasping with Sim-to-Real Transfer based Reinforcement Learning. *arXiv* **2018**, arXiv:1803.04996.
6. Salvato, E.; Fenu, G.; Medvet, E.; Pellegrino, F.A. Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning. *IEEE Access* **2021**, *9*, 153171–153187. [CrossRef]
7. Nagabandi, A.; Clavera, I.; Liu, S.; Fearing, R.S.; Abbeel, P.; Levine, S.; Finn, C. Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning. *arXiv* **2019**, arXiv:1803.11347.
8. Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 23–30.
9. Volpi, R.; Larlus, D.; Rogez, G. Continual Adaptation of Visual Representations via Domain Randomization and Meta-learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 4441–4451.
10. Jiang, Y.; Zhang, T.; Ho, D.; Bai, Y.; Liu, C.K.; Levine, S.; Tan, J. SimGAN: Hybrid Simulator Identification for Domain Adaptation via Adversarial Reinforcement Learning. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 2884–2890.
11. Ho, D.; Rao, K.; Xu, Z.; Jang, E.; Khansari, M.; Bai, Y. RetinaGAN: An Object-aware Approach to Sim-to-Real Transfer. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 10920–10926.
12. Arndt, K.; Hazara, M.; Ghadirzadeh, A.; Kyrki, V. Meta Reinforcement Learning for Sim-to-real Domain Adaptation. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 2725–2731. [CrossRef]
13. Traoré, K.R.; Caselles-Dupré, H.; Lesort, T.; Sun, T.; Rodríguez, N.D.; Filliat, D. Continual Reinforcement Learning deployed in Real-life using Policy Distillation and Sim2Real Transfer. *arXiv* **2019**, arXiv:1906.04452.
14. Zhang, F.; Leitner, J.; Milford, M.; Upcroft, B.; Corke, P. Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control. *arXiv* **2015**, arXiv:1511.03791.
15. James, S.; Johns, E. 3D Simulation for Robot Arm Control with Deep Q-Learning. *arXiv* **2016**, arXiv:1609.03759.

16. Liu, N.; Cai, Y.; Lu, T.; Wang, R.; Wang, S. Real–Sim–Real Transfer for Real-World Robot Control Policy Learning with Deep Reinforcement Learning. *Appl. Sci.* **2020**, *10*, 1555. [CrossRef]
17. Rao, K.; Harris, C.; Irpan, A.; Levine, S.; Ibarz, J.; Khansari, M. RL-CycleGan: Reinforcement learning aware simulation-to-real. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 11154–11163. [CrossRef]
18. Du, Y.; Watkins, O.; Darrell, T.; Abbeel, P.; Pathak, D. Auto-Tuned Sim-to-Real Transfer. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 1290–1296.
19. Obando-Ceron, J.S.; Castro, P.S. Revisiting Rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021.
20. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
21. Van Hasselt, H.; Guez, A.; Silver, D. Deep Reinforcement Learning with Double Q-Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30. [CrossRef]
22. Behzadan, V.; Munir, A. Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. *arXiv* **2017**, arXiv:1701.04143.
23. Spears, T.; Jacques, B.; Howard, M.; Sederberg, P. Scale-invariant temporal history (SITH): Optimal slicing of the past in an uncertain world. *arXiv* **2017**, arXiv:1712.07165.
24. Cai, W.; Wen, X.; Wang, S.; Wang, L. A real-time detection method of building energy efficiency based on image processing. *J. Vis. Commun. Image Represent.* **2019**, *60*, 295–304. [CrossRef]
25. Bui, H.M.; Lech, M.; Cheng, E.; Neville, K.; Burnett, I.S. Using grayscale images for object recognition with convolutional-recursive neural network. In Proceedings of the 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), Ha-Long, Vietnam, 27–29 July 2016; pp. 321–325. [CrossRef]
26. Zhao, W.; Queralta, J.P.; Qingqing, L.; Westerlund, T. Towards Closing the Sim-to-Real Gap in Collaborative Multi-Robot Deep Reinforcement Learning. In Proceedings of the 2020 5th International Conference on Robotics and Automation Engineering (ICRAE), Singapore, 20–22 November 2020; pp. 7–12.
27. Chen, X.; Ling, J.; Wang, S.; Yang, Y.; Luo, L.; Yan, Y. Ship detection from coastal surveillance videos via an ensemble Canny-Gaussian-morphology framework. *J. Navig.* **2021**, *74*, 1252–1266. [CrossRef]
28. Tsai, Y.-Y.; Xu, H.; Ding, Z.; Zhang, C.; Johns, E.; Huang, B. DROID: Minimizing the Reality Gap Using Single-Shot Human Demonstration. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3168–3175. [CrossRef]
29. Zhou, S.; Pereida, K.; Zhao, W.; Schoellig, A.P. Bridging the Model-Reality Gap With Lipschitz Network Adaptation. *IEEE Robot. Autom. Lett.* **2022**, *7*, 642–649. [CrossRef]