applied sciences

MDPI

Article

# Real–Sim–Real Transfer for Real-World Robot Control Policy Learning with Deep Reinforcement Learning †

**Naijun Liu [1,2], Yinghao Cai [1,*], Tao Lu [1,*], Rui Wang [1,3] and Shuo Wang [1,2,4,*]**

1 State Key Laboratory of Management and Control for Complex Systems, Institute of Automation Chinese Academy of Sciences, Beijing 100190, China; liunaijun2016@ia.ac.cn (N.L.); rwang5212@ia.ac.cn (R.W.)
2 University of Chinese Academy of Sciences, Beijing 100190, China
3 Guangdong Provincial Key Lab of Robotics and Intelligent System, Shenzhen Institutes of Advanced Technology Chinese Academy of Sciences, Shenzhen 518055, China
4 Center for Excellence in Brain Science and Intelligence Technology of the Chinese Academy of Sciences, Shanghai 200031, China
* Correspondence: yinghao.cai@ia.ac.cn (Y.C.); tao.lu@ia.ac.cn (T.L.); shuo.wang@ia.ac.cn (S.W.); Tel.:+86-10-82544525 (Y.C. & T.L. & S.W.)
† This paper is an extended version of our paper published in the 32nd Chinese Control and Decision Conference (CCDC).

Received: 12 January 2020; Accepted: 20 February 2020; Published: 25 February 2020


**Abstract:** Compared to traditional data-driven learning methods, recently developed deep reinforcement learning (DRL) approaches can be employed to train robot agents to obtain control policies with appealing performance. However, learning control policies for real-world robots through DRL is costly and cumbersome. A promising alternative is to train policies in simulated environments and transfer the learned policies to real-world scenarios. Unfortunately, due to the reality gap between simulated and real-world environments, the policies learned in simulated environments often cannot be generalized well to the real world. Bridging the reality gap is still a challenging problem. In this paper, we propose a novel real–sim–real (RSR) transfer method that includes a real-to-sim training phase and a sim-to-real inference phase. In the real-to-sim training phase, a task-relevant simulated environment is constructed based on semantic information of the real-world scenario and coordinate transformation, and then a policy is trained with the DRL method in the built simulated environment. In the sim-to-real inference phase, the learned policy is directly applied to control the robot in real-world scenarios without any real-world data. Experimental results in two different robot control tasks show that the proposed RSR method can train skill policies with high generalization performance and significantly low training costs.

**Keywords:** robot; policy learning; reality gap; simulated environment; deep reinforcement learning

## 1. Introduction

Over the past decades, robots have been gradually applied in various fields, with the expectation of completing more control tasks for human beings. Traditional programming methods can achieve the goal of performing certain tasks with the assumption that environments are known and structured [1]. However, robots often encounter working scenarios that are complicated and unpredictable in the real world. As a result, significant research attention has been given to data-driven learning methods [2,3], which avoid some of the challenges of analytic formulations and endow the learned policies with generalization capability. Recently, deep reinforcement learning (DRL) [4], which combines the reinforcement learning (RL) [5] method with deep neural networks has achieved great success in areas such as video games [6] and

the board game Go [7]. Inspired by this, many works try to apply DRL algorithms in training robots to obtain control policies in unstructured environments, which shows appealing performance [8]. However, DRL methods typically require huge amounts of training samples and large-scale random explorations, which bring mechanical wear and tear to the hardware of robots. As collecting training data on real-world robots is costly, potentially unsafe, and time-consuming, learning control policies for real-world robots can be difficult and tedious.

One promising method is to train control policies in simulated environments where data generation is safe, convenient, and linvolves a ower cost, and then to transfer the learned policies to the real world. However, it is laborious to construct simulated environments similar to real-world working scenarios, especially with high fidelity. Consequently, the policies trained in simulated environments usually cannot directly work well in the real world due to the reality gap (discrepancies between simulated and real-world environments). Although lots of approaches have been proposed to bridge the reality gap, such as domain randomization (DR) [9] and domain adaptation (DA) [10], bridging the reality gap is still a challenging problem.

To train control policies for real-world robots with high generalization capability, and to greatly reduce the training cost, in this work we propose a real–sim–real (RSR) transfer method that includes a real-to-sim training phase and a sim-to-real inference phase. In the real-to-sim training phase, a simplified task-relevant simulated environment is automatically constructed based on the semantic information of the real-world scenario and coordinate transformation. The control policies are trained with the DRL method in the built environment. In the sim-to-real inference phase, the trained policies are directly transferred to the real-world scenarios. Experimental results show that the proposed RSR method can train control policies for real-world robots with promising generalization performance and significantly low training costs.

In summary, the main contributions of this paper are listed as follows:

(1) We present a new learning paradigm to train control policies for real-world robots with the DRL method. The learning pipeline is divided into a real-to-sim training phase and a sim-to-real inference phase, which trains robot control policy with a higher generalization capability and lower costs.
(2) The proposed method automatically constructs a task-relevant simulated environment for policy learning based on semantic information of real-world working scenarios and coordinate transformation, which avoids the challenging problem of manually creating the simulated environments with high fidelity, endowing the policy learning process with high efficiency.
(3) The proposed method directly employs the trained policy in real-world scenarios without any real-world training data or fine-tuning.

The rest of this document is organized as follows. In Section 2, previous research in this field is summarized. Section 3 describes the details of the proposed method. Section 4 shows the experiments and results. Finally, Section 5 presents the conclusions.

## 2. Related Work

### 2.1. Robot Control Policy Learning

Data-driven learning algorithms are widely employed to train control policies, which can be classified into supervised learning methods, reinforcement learning methods, and recently developed deep reinforcement learning methods. The supervised learning methods take the state–action pairs of demonstration data as the training samples to learn mapping relationships between the states and actions [11], which have been successfully deployed in manipulation tasks [12,13], driving [14], and navigation [15]. However, generally, the policies learned with supervised learning methods are deeply influenced by the quality of demonstrations. Typically collecting high-quality demonstration data, especially in the field of robots, is not a trivial task [16,17].

Reinforcement learning methods train the robot agents to obtain optimal policies through trial and error [18–22]. Reinforcement learning has led to many successes in the domain of robot control when low-dimensional state space or action space is available [23,24]. However, reinforcement learning shows limited success in continuous cases.

Deep reinforcement learning methods combining reinforcement learning with deep neural networks have shown great potential in addressing high-dimensional and continuous action-state space for robot control policy learning. Deep Q-network (DQN) [6] has been implemented to train reaching skill in 2D space for three-link robots [25,26]. In addition, deep deterministic policy gradient (DDPG) [27], trust region policy optimization (TRPO) [28], asynchronous advantage actor-critic (A3C) [29], generalized advantage estimation (GAE) [30], and proximal policy optimization(PPO) [31] have also been implemented in certain robot simulated control tasks, such as stacking blocks, hopping, or walking. Guided policy search (GPS) [32] is a rare method that can directly train control policies on real-world robots. Although deep reinforcement learning methods show high potential for control policy learning, due to large amounts of training data notoriously to collect, acquiring manipulation skill policies for real-world robots through DRL is time-consuming and cumbersome.

### 2.2. Sim-to-Real Transfer

The simulated environment is an appealing alternative to real-world scenarios for policy learning. However, the reality gap also introduces new challenges that have to be solved to make the trained policies be effectively applied in real-world scenarios. A number of recent works have explored different strategies for policy learning in the context of robot control.

One natural way is to make simulated environments closely match the real world by using high-quality rendering. Some researchers create visually-realistic simulated environments for 3-link robot reaching skill learning [25,26], or for 7-DOF robot arm grasping skill learning [33], hoping that the trained policies exhibit similar behavior in the real world as its simulated counterpart, showing limited success. Others [34,35] use simulated depth images, which abstract away appearance properties of objects, and then employ the learned policy in the real world with a calibrated fixed depth camera. Unfortunately, a simulated environment rarely models the real world perfectly, and implementing the policies trained in an imperfect simulation model can yield a poor real-world performance. Unlike these approaches, our method allows the use of low-quality renderers that are not carefully matched to real-world scenarios, which is beneficial for low-cost policy learning.

Other works explore using domain adaptation to bridge the reality gap. Domain adaptation allows a learning model trained with data from a simulation domain to generalize to a real-world domain [36]. Stein et al. utilizes cycleGAN to convert each synthetic image to the realistic style one [37]. Cutler et al. uses simulation data as a prior to train control policies, which decreases the real-world samples [38]. Transferring synthetic images from simulator to adapted images similar to real-world ones is also adopted [39]. Applying progressive networks [40], which share features between simulated environments and real world, enable the learning of a manipulation policy. Domain adaptation is an important tool for addressing the reality gap, but in contrast to these approaches, ours requires no additional training on real-world data.

Several works have shown the success of exploring the idea of domain randomization to bridge the reality gap. Policies learned in a simulator with varied 3D scenes and textures can be applied successfully to real-world quadrotor flight [41]. Similarly, randomizing the texture of objects, lighting conditions, and camera positions in the simulated environments during training is proposed [9], with the aim that models learned in simulation would generalize to real-world scenarios with no additional training. This involves manually adjusting the simulated environment to roughly match the appearance and dynamics of the laboratory setup, and then relying on domain randomization of only the camera position and orientation [42]. As the dynamics of a simulated robot may differ from its real-world counterpart, domain randomization is also explored in dynamics [43,44].Other works

also explore combining domain randomization with domain adaptation [45] for policy training on reaching tasks.

Unlike these approaches of manually designing a simulated model, which are grueling and time-consuming, the proposed RSR transfer method can automatically construct a task-relevant simulated environment based on semantic images of real-world scenes and coordinate transformation, which guarantees that the constructed simulated environment resembles its real-world counterpart with respect to policy training. In addition, our approach does not require any real-world training and attempts to directly apply policies learned in simulation to a real-world robot, without the burden of requiring human interactions during the training process.

## 3. Method

The background of classic reinforcement learning is a Markov decision process (MDP), defined by a tuple $(\mathbf{S}, \mathbf{A}, \pi, r, P, \gamma)$, where $\mathbf{S}$ is a state set, $\mathbf{A}$ is an action set, $\pi : \mathbf{S} \to \mathbb{R}$ is a policy, $r : \mathbf{S} \times \mathbf{A} \to \mathbb{R}$ is a reward function, $P : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to \mathbb{R}$ is a transition dynamic, and $\gamma \in (0, 1)$ is a discount factor. When the agent interacts with the environment using policy $\pi$, a trajectory sequence $\tau : \{s_0, a_0, r_0, s_1, a_1, r_1, \cdots, s_T, a_T, r_T\}$ is rolled out, where $T$ is the length of $\tau$. Discounted accumulated rewards $R(\tau)$ can be written as:

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t(s_t, a_t), 0 < \gamma < 1, \tag{1}$$

where $a_t \sim \pi(a_t|s_t)$, $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$, and "|" is a symbol for conditional probability. Value function $V^\pi(s_t)$, state-action value function $Q^\pi(s_t, a_t)$, and advantage function $A^\pi(s_t, a_t)$ are defined as:

$$V^\pi(s_t) = \mathrm{E}[\sum_{k=t}^{T} \gamma^{k-t} r_t(s_t, a_t)|s = s_t; \pi], \tag{2}$$

$$Q^\pi(s_t, a_t) = \mathrm{E}[\sum_{k=t}^{T} \gamma^{k-t} r_t(s_t, a_t)|S = s_t, A = a_t; \pi], \tag{3}$$

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t). \tag{4}$$

The policy is optimized by maximizing the accumulated rewards $R(\tau)$.

Most common robot control tasks such as manipulation or navigation tasks require a robot to reach a desired state from an initial state. As a result, in this paper, we focus robot control policy learning on different forms of reaching tasks in relatively complicated environments with obstacles, which are still challenging and also the stepping stones to more complex tasks. Figure 1 shows the learning pipeline of our proposed method, which includes a real-to-sim training phase and a sim-to-real inference phase, as shown in Figure 1a,b, respectively. To make the content of this paper more concise and compact, by default, we mainly take the manipulation task as an example to illustrate our method. At the real-to-sim training phase, firstly, a semantic image is segmented from an RGB image of a real-world working scenario. Coordinate transformation maps each pixel position from the image coordinate system to the robot coordinate system. Then, a task-relevant simulated environment is generated based on the semantic information and coordinate transformation. Finally, a control policy is learned with the DRL method in the constructed simulated environment. At the sim-to-real inference phase, simulated-like synthetic images are generated based on the semantic images of the real wold. The trained policy takes the synthetic images as input and outputs actions that thus directly control the real-world robot to perform the desired task.

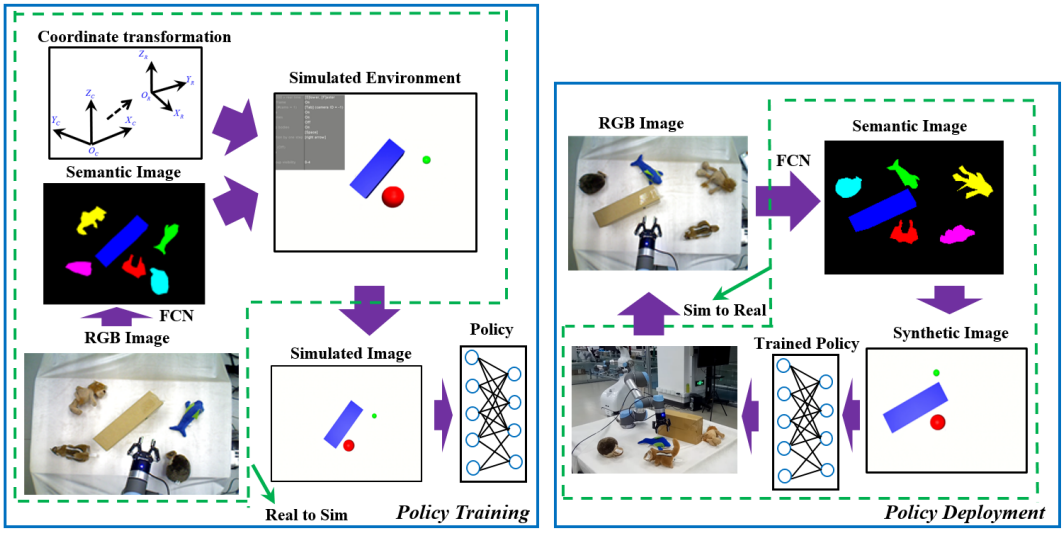



**Figure 1.** Illustration of the proposed real–sim–real (RSR) transfer learning pipeline. (**a**) Real-to-sim training phase. A task-related simulated environment is generated based on coordinate transformation and semantic images of real-world robot working scenarios. A policy network is trained in the constructed simulated environment with the deep reinforcement learning (DRL) method. (**b**) Sim-to-real inference phase. The trained policy takes simulated-like synthetic images as input and outputs actions to control real-world robots via an ROS (robot operation system).

### *3.1. Generating a Simulated Environment*

An RGB image $I_{rgb}$ and a depth image $I_{dpt}$ of a robot working scenario are captured from an RGB-D camera. As is shown in Figure 1a, a semantic image $I_{sem}$ is segmented from $I_{rgb}$ based on fully convolutional networks (FCN) [46]. To conveniently and effectively construct a task-relevant simulated environment for policy learning, some simplifications are made for semantic image $I_{sem}$ to create a simulated environment, as is shown in Figure 2. The target object is simplified to its geometry center. The robot (for navigation task) or gripper (for manipulation task) is simplified to be a solid ball, the center and diameter of which are calculated from its semantic pixel region contour. The obstacles are completely preserved, and other irrelevant objects that cannot be obstacles are ignored. Given depth image $I_{dpt}$, our method transforms each pixel position $[u_i, v_i]^T$ of the RGB image from the image coordinate system to the robot coordinate system with the following coordinate transformation equation:

$$p_{ri} = \begin{bmatrix} x_{ri} \\ y_{ri} \\ z_{ri} \end{bmatrix} = Rz_{ci} M_{in}^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} + T, \tag{5}$$

where $i = 1, 2, \cdots, T$, $p_{ri}$ is the transformed position under the robot coordinate system, $M_{in}$ is the camera inner parameter matrix, $z_{ci}$ is the depth value with respect to the pixel position $[u_i, v_i]^T$, and $R$ and $T$ are the calibrated rotation matrix and transformation vector from the camera coordinate system to the robot coordinate system. As a result, we obtain the pose information of the corresponding objects under the robot coordinate system.

Consequently, we get a task-relevant simulated environment, which is an abstraction of the real-world scene, and meanwhile keep the information related to training the desired control policy. In the policy training period, the solid red ball corresponding to the robot (or gripper) represents the virtual agent, the solid green circle corresponding to the target object denotes the target position, the blue area corresponds to the obstacle area being unreachable for the virtual agent, and the black area corresponds to the background and irrelevant objects area being reachable for the virtual agent.

The initial positions of the virtual agent and target object, and the shape and number of obstacles can be changed randomly in the simulated environment.

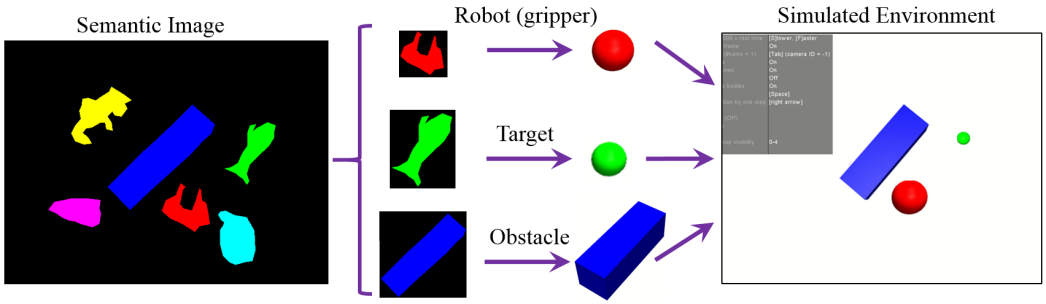



**Figure 2.** Illustration of generating a task-relevant simulated environment. The target object is simplified to its geometry center of the corresponding regions in the semantic image. The robot (or gripper) is equivalent to a solid ball, the center and diameter of which are calculated from its semantic pixel region contour. The obstacles are completely preserved, and other irrelevant objects that cannot be objects are ignored.

### 3.2. Policy Network

The designed policy network is inspired by [32], including three convolutional layers and two fully connected layers, as is shown in Figure 3. The policy network takes simulated image $I_{s,t}$ captured from the simulated environment at current time step $t$ together with simulated images $I_{s,t-1}$ and $I_{s,t-2}$ at the time steps $t-1$ and $t-2$ as input. The simulated images are resized from the raw image size $(640 \times 480)$ to be $(240 \times 240)$. The output of the network is the mean $\mu_\theta(I_{s,t})$ and variance $\sigma_\theta(I_{s,t})$ of a Gaussian policy $\pi_\theta(\cdot|I_{s,t}) = N(\mu_\theta(I_{s,t}), \sigma_\theta(I_{s,t}))$, where $\theta$ represents the parameters of the policy neural network.

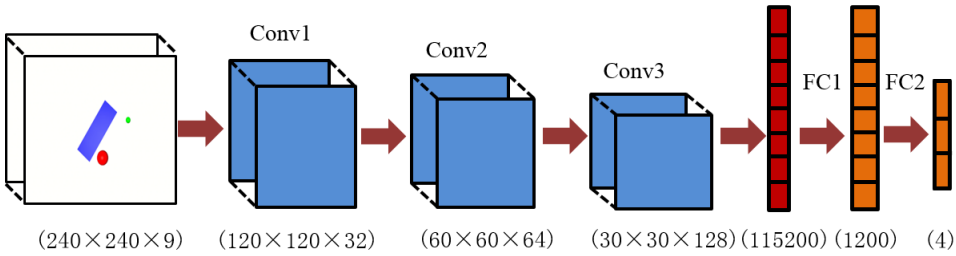



**Figure 3.** Architecture of the designed neural network policy. Two stride steps and a filter size of $3 \times 3$ are used for all three convolution networks. The ReLU nonlinear activation function is used throughout, with no pooling and no dropout techniques applied. The input is three RGB images from a simulated environment downsized to $(240 \times 240)$ with 9 channels, and the output is action.

### 3.3. Policy Training

At time step $t$ , the robot agent takes an action $a_t$ according to current state $I_{s,t}$ and policy $\pi_\theta$, receives reward $r_t$, and moves to the next state $I_{s,t+1}$. Repeating the above procedure, an episode trajectory is obtained $\tau : \{I_{s,0}, a_0, r_0, \cdots, I_{s,t}, a_t, r_t, \cdots, I_{s,T}\}$. The reward function $r_t$ is set to be

$$r_t = \begin{cases} -10, & \text{encountering obstacle,} \\ 0, & d \leq \delta, \\ -d, & \text{otherwise,} \end{cases} \tag{6}$$

where $d = \|x - x^*\|$ is the Euclidean distance between the robot (or gripper) position $x$ and the target object center $x^*$, $\delta$ is the threshold to determine whether the agent reaches the target position

($\delta = 1$ pixel). When encountering obstacles, the agent receives a reward of $-10$. We adopt the DRL method of proximal policy optimization (PPO) [31] to maximize a surrogate objective $L^{clip}(\theta)$:

$$L^{clip}(\theta) = E[min(\frac{\pi_\theta(a_t|I_{s,t})}{\pi_{\theta old}(a_t|I_{s,t})}\widehat{A}_t, k(\varepsilon, \widehat{A}_t)], \tag{7}$$

where

$$k(\varepsilon, \widehat{A}_t) = \begin{cases} (1+\varepsilon)\widehat{A}_t, \widehat{A}_t \geq 0 \\ (1-\varepsilon)\widehat{A}_t, \widehat{A}_t < 0 \end{cases}. \tag{8}$$

$\varepsilon = 0.2$, $t$ specifies the time index in $[0, T]$, $\pi_{\theta old}$ denotes the old policy before the update, and $\widehat{A}_t$ is the estimated advantage function $A_t$,

$$\begin{aligned} \widehat{A}_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \\ \gamma^{T-t} V_\varphi(I_{s,T}) - V_\varphi(I_{s,t}), \end{aligned} \tag{9}$$

where $V_\varphi(I_{s,t})$ is the estimated value function for state $I_{s,t}$, and the parameter $\varphi$ is updated by regression on mean-squared error,

$$\varphi = \arg\min E(\sum_\tau \sum_{t=0}^{T} \left(V_\varphi(I_{s,t}) - R(t)\right)^2). \tag{10}$$

The policy parameter $\theta$ is updated by

$$\theta = \theta + \alpha \nabla_\theta L^{clip}(\theta), \tag{11}$$

where $\alpha$ is the learning rate.

### *3.4. Deploying the Trained Policy*

The sketch of deploying the policy trained from the simulated environment to the real-world scenario is shown in Figure 1b. Similar to the policy training period, semantic image $I_{sem}$ is segmented from the captured RGB image. Our method then synthesizes simulated-like images $I_{syn}$ in low-fidelity based on a segmented image respecting the following rules: the target object is simplified to be its geometry center; the robot (or gripper) is equivalent to a solid circle, the center and diameter of which are calculated from the semantic pixel region contour; the obstacles are completely preserved, and other irrelevant objects that cannot be obstacles are ignored.

Similar to the training phase, at time step $t$ of the policies employed period, the trained policy takes synthetic image $I_{syn,t}$ together with synthetic images $I_{syn,t-1}$ and $I_{syn,t-2}$ at time $t-1$ and $t-2$ as input and outputs actions that directly control the real-world robot. As a result, we do not fine-tune the trained policy with real-world training data in the inference phase. The only additional step is to convert the real-world images to simulated-like synthetic images, which is efficient and inexpensive for policy learning.

The fully detailed algorithm is shown in Algorithm 1.

### *3.5. Performance Evaluation*

The performance of the learned policy is evaluated in a real-world scenario in terms of success rate $S_{rate}$, which specifies the ratio of the times of successfully achieving the desired task within the allowed error $\delta$ to all testing times $N$ consumed.

$$S_{rate} = \frac{\sum_i^N \hbar(d_e^i \leq \delta)}{N}, \tag{12}$$

where $\hbar(\cdot)$ is a indicator function outputting 1 when taking *True* as input and outputting 0 when taking *False* as input, and $d_e^i = \left\| p_f - p_d \right\|$ is the distance error measured by the Euclidean distance between the target position $p_d$ and the final robot (or gripper) position $p_f$ at the end of the $i$th episode.

**Algorithm 1** RSR transfer method

**Real-to-sim training phase:**

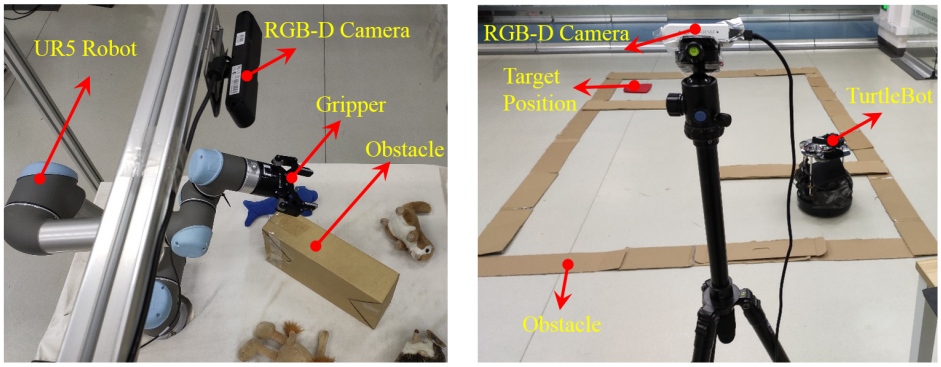
Capture RGB image $I_{rgb}$ and depth image $I_{dpt}$ from RGB-D camera.
Obtain semantic image $I_{sem}$ based on FCN.
Construct a task-related simulated environment with $I_{sem}$ and coordinate transformation.
Design a policy network.
Train policy with PPO in the constructed simulated environment.
**for** $k = 1, 2, \cdots, do$ **do**
    Collect trajectory $\tau$.
    Update policy parameter $\theta$ by maximizing the surrogate objective $L^{clip}(\theta)$.(Equation (7))
    Fit value function $V_\varphi$.(Equation (10))
**end for**
Until policy converges.

**Sim-to-real inference phase:**

**for** $k = 1, 2, \cdots, T$ **do**
    Semantic image $I_{sem}$ is segmented from the captured real-world RGB image.
    Synthesize simulated-like images $I_{syn}$.
    The trained policy takes synthetic images as input, and output actions controlling the real-world robot.
**end for**
Until finish the target task.

## 4. Experiments and Results

To evaluate the proposed RSR method, experiments were carried out on two designed tasks: a UR5 robot manipulation task and TurtleBot navigation task, as shown in Figure 4, respectively. The first task was learning a skill policy to control the robot gripper to reach a target object, avoiding obstacles in 3D space. The second one was to train the TurtleBot to navigate from a random starting position to a random goal position in 2D space, without obstacle collision as well. A real-world RGB-D camera was visually calibrated to match the position and orientation of the simulated camera for each task respectively.




(a) (b)

**Figure 4.** Two designated tasks are evaluated for our method. (**a**) Manipulation task. A policy is trained to control a UR5 robot gripper to reach the target object in 3D space and avoid obstacles (yellow box). (**b**) Navigation task. Learning a policy navigates the TurtleBot to the target position without obstacle collision.

### *4.1. Semantic Segmentation of Robot Working Scenarios*

In this work, for the manipulation task, the objects situated in our robot working scenarios were classified into background, robot gripper, obstacle, toy dolphin, toy hedgehog, toy squirrel, and toy lion. For the navigation task, the objects were classified into background, robot, obstacle, and target object. We collected RGB images of the robot working scenarios from the real-world camera. To make the training data for semantic segmentation, each pixel of the RGB images was labeled with one of the above categories.

The FCN neural network was based on VGG-16 [47], which has a wide availability of pre-trained weights. We generated a training set of 200 samples and validation set of 50 samples for each task. The semantic segmentation network was converged after 20 iteration steps using the SGD (stochastic gradient descent) optimization method with a batch size of 32 images. The semantic segmentation results are shown in Figure 5. We also found that the segmentation module works well in scenarios with objects overlapping with each other. Moreover, we adopted a dilation and erosion technique to filter noise from each semantic image, which we found to be beneficial to obtain the centroids of robot (or gripper) or target object.

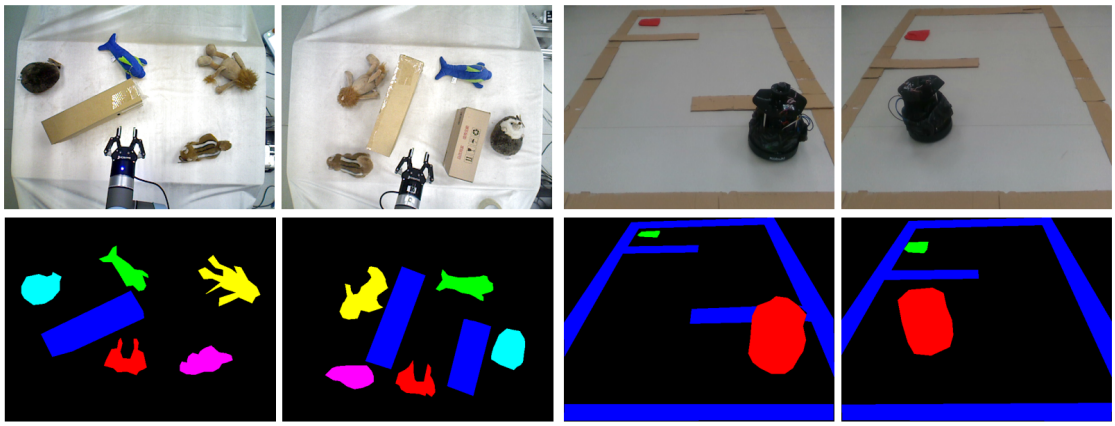
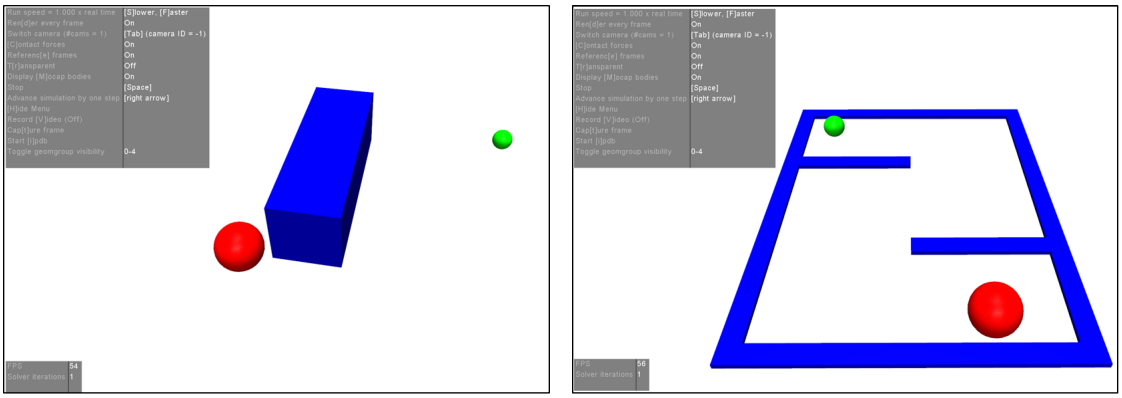


**Figure 5.** Sematic segmentation results of robot working scenarios. The first row shows the RGB images, and the second row shows their corresponding semantic images. The left two are for the manipulation task, and the right two are for the navigation task. Best viewed in color.

### *4.2. Policy Learning*

The task-relevant simulated environment was generated automatically based on our proposed method in the Mujoco physics engine [48] interfaced with OpenAI Gym [49], as is shown in Figure 6. For the manipulation task, the action dimension is 3, which moves the gripper in 3D space. For the navigation task, the action dimension is 2, which controls the TurtleBot's navigation in 2D space.



(a) (b)

**Figure 6.** Task-relevant simulated environments generated based on our proposed method. (**a**) Manipulation task. (**b**) Navigation task. The blue objects are the obstacles, solid red balls are the virtual agents, and green balls denote the target objects (positions). The gray areas illustrate control information of the constructed environment. Best viewed in color.

We compared our method against several baseline methods.

(1) Transfer-RGB: direct training policy with simulated RGB images and using real-world RGB images in the inference period.
(2) Transfer-Depth: training policy with simulated depth images and using real-world depth images in the inference period.
(3) DR (domain randomization): In the policy training period, the mesh of the objects in the simulated environment is randomly chosen from 50 textures. The camera position and orientation remain fixed, which matches the real-world scenario. The trained policy is directly employed in the real-world scenario in the policy inference period.
(4) DA (domain adaptation): First, the training policy is implemented in the simulated environment and then the trained policy is fine-tuned using the same amount of real-world training data as in the simulated environment.

For the Transfer-RGB, Transfer-Depth, DR, and DA methods, we built a simulated environment similar to our real-world robot working scenario in Gazebo simulator (https://gazebosim.org). The domain randomization technique refers to [9] (https://github.com/neka-nat/gazebo_domain_randomization). All of the methods share the same neural networks. We initialized the convolutional layers of the policy networks by the weights pre-trained on ImageNet. The policy networks are trained with TensorFlow (https://www.tensorflow.org) on NVIDIA GTX1080. Table 1 summarizes the parameters used in our experiments.

**Table 1.** Parameters in our experiments.

| Parameter | Value |
|---|---|
| Learning rate for policy $\alpha$ | $3.0 \times 10^{-4}$ |
| Learning rate for value function $\beta$ | $1.0 \times 10^{-3}$ |
| Length of horizon $T$ | 100 (Manipulation); 200 (Navigation) |
| Discount $\gamma$ | 0.99 |
| Rollouts per iteration | 20 |
| Batch size | 32 |
| Optimization method | Adam [50] |

In the policy training phase, to evaluate the success rates of the trained policies for the manipulation task, we randomly chose one of the toys (toy dolphin, toy hedgehog, toy squirrel, and toy lion) as the target object to be randomly put in front of the UR5 robot, within its workspace.

The success rates were evaluated by performing the manipulation task 20 times with the robot gripper in different starting positions and the target object in different goal positions every 30 policy iteration steps. For the navigation task, the navigation scenario was a manually designed rectangular area with a length of 4.5 m and a width of 4.0 m.The success rates were calculated by conducting the navigation task 20 times with different initial positions and target positions. The success rates were also evaluated every 30 policy iteration steps. To test the generalization performance of the trained policy, we randomly changed the robot (or gripper) starting position, target object position, obstacle position, and the number of obstacles in real-world scenarios.

We trained three different instances of each algorithm with different random seeds. Figure 7a,b illustrates the learning curves of our proposed method and the baseline methods applied in the manipulation task and navigation task, respectively. The solid curves corresponds to the mean success rates and the shaded region to the minimum and maximum success rates over the three trials. Table 2 summarizes the average success rates of the final trained polices. The proposed method achieves an average success rates of 89% in the manipulation task and 93% in the navigation task. Compared to the DR and DA methods, the results on two designed real-world tasks show that our proposed learning pipeline shows a better accuracy performance and higher-generalization capability. We find that the policy trained in a simulated environment with RGB images cannot be successfully deployed in real-world scenarios, confirming that the reality-gap has a significant harmful influence on policies directly transferred from simulated environments to the real world. The policy trained with depth images also demonstrates poor performance. Although the generated simulated-like synthetic images do not contain rich information like the real-world ones, the experimental results show that images rendered in low-fidelity with our method provide useful information for policy learning.

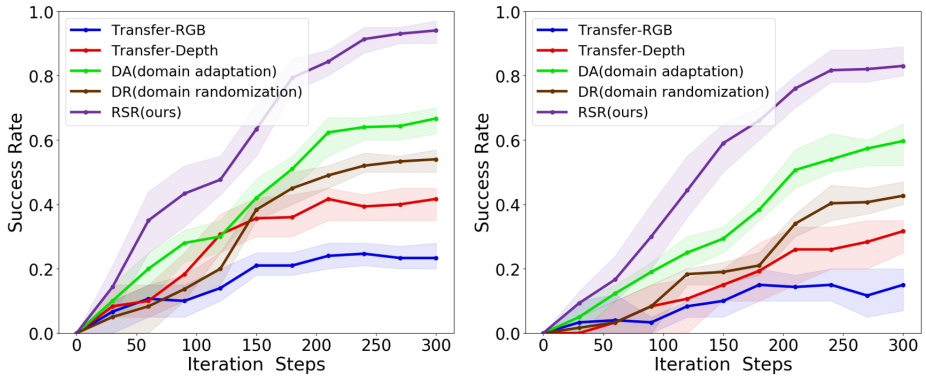



**Figure 7.** Learning curves for the performance of the policies trained with different methods. (**a**) Manipulation task. (**b**) Navigation task. Compared to the baseline methods, our proposed method shows a better performance.

**Table 2.** Average success rates of the final trained policies employed in real-world scenarios.

| Methods | Manipulation Task | Navigation Task |
|---|---|---|
| Transfer-RGB | 24% | 15% |
| Transfer-Depth | 42% | 32% |
| DR (domain randomization) | 67% | 61% |
| DA (domain adaptation) | 53% | 43% |
| RSR(ours) | **93**% | **83**% |

Another advantage of our learning paradigm over existing methods is that it is efficient and low-cost, due not only to it not needing real-world data or fine-tuning for real-world policy learning, but also in constructing the simulated environment for policy learning.

Figures 8 and 9 show the frames of the final trained policies deployed on the manipulation task and the navigation task in the constructed simulated environment, respectively. Our proposed method succeeds in learning these two designed tasks in simulated environments.

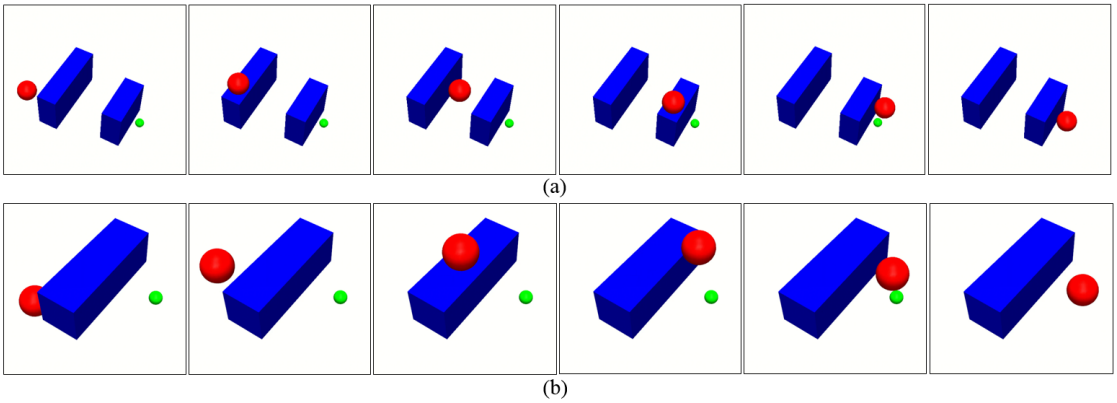


**Figure 8.** Frames captured from the final policies trained with our proposed method deployed on manipulation tasks in simulated environments. (**a**) two obstacles. (**b**) one obstacle. Best viewed in color.

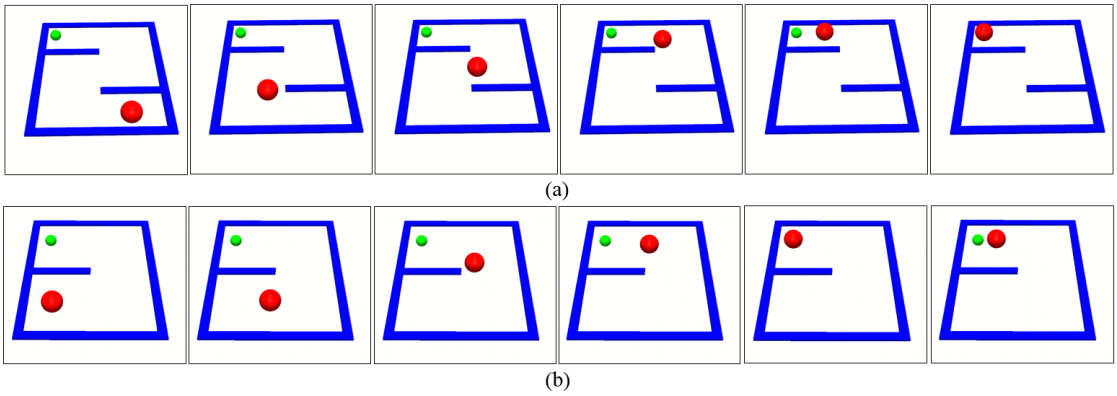


**Figure 9.** Frames captured from the final policies trained with our proposed method deployed on navigation task in simulated environments. (**a**) two obstacles. (**b**) one obstacle. Best viewed in color.

Figures 10 and 11 show the frames of the final trained policies employed on two designed tasks in real-world environments, respectively.

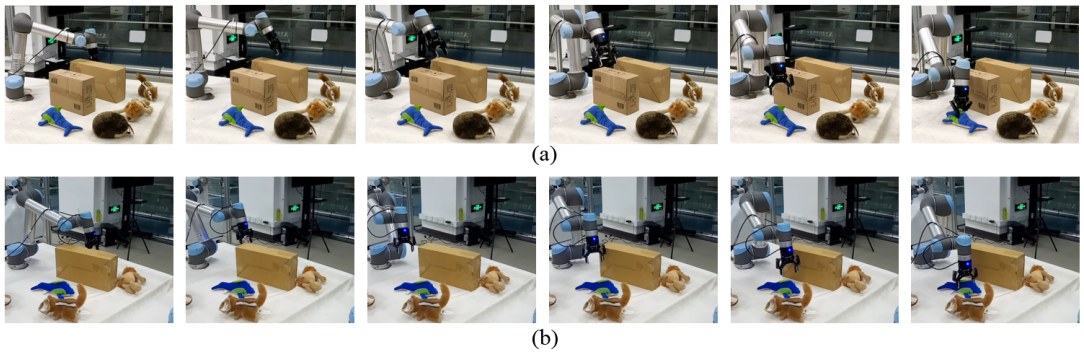


**Figure 10.** Frames of the final policies trained with our method employed on the manipulation task in real-world working scenarios. The robot gripper reaches the target object (toy dolphin) in 3D space and avoiding obstacles. (**a**) two obstacles. (**b**) one obstacle.

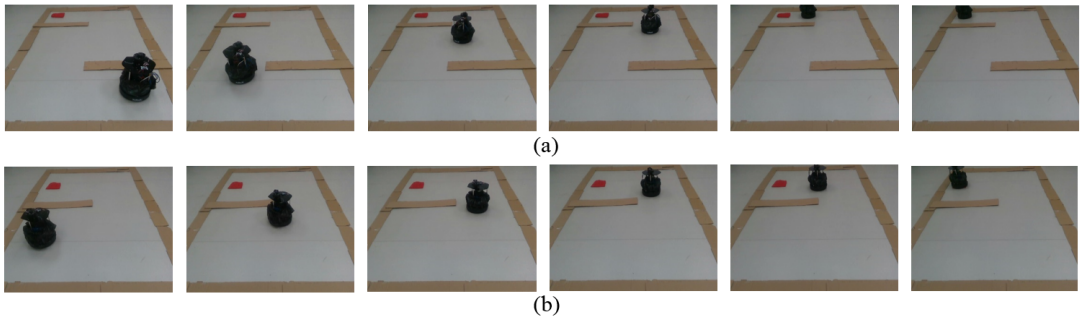


**Figure 11.** Frames of the final policies trained with our method employed on the navigation task in real-world working scenarios. The TurtleBot navigates to the target position without obstacle collision. (**a**) two obstacles. (**b**) one obstacle.

## 5. Conclusions

In this paper, we study the possibility of directly transferring the policies trained in simulated environments to the real world with high generation capability and low costs. We proposed a novel real–sim–real (RSR) transfer method for control policy learning in real-world robots. In the real-to-sim training phase, a task-relevant simulated environment is automatically constructed based on semantic information of real-world working scenarios and coordinate transformation, and then policies are learned in the built simulated environment with the DRL method. In the sim-to-real inference phase, the trained policy is directly employed in the real world. As real-world scenarios are usually complicated and unstructured, the DRL method shows great potential for developing skill policies to be well employed in such environments. The experimental results show that our proposed method can effectively and efficiently learn control policies for real-world robots using the DRL method. The policies trained with our method show high generalization capability and low costs.

In future works, we intend to extend the training scenarios to richer repertoire tasks that are more common in real life. In addition, to improve the performance of the trained policies, we would incorporate more modalities such as robot state information or haptic sensor information for policy learning. Another direction is combining our RSR method with domain adaptation or domain randomization methods.

**Author Contributions:** Conceptualization, N.L. and T.L.; methodology, N.L. and Y.C.; software, N.L.; validation, N.L., R.W., T.L., and Y.C.; formal analysis, N.L.; investigation, N.L.; resources, N.L.; writing original draft preparation, N.L.; writing review and editing, T.L., Y.C., R.W., and S.W.; visualization, N.L.; supervision, S.W.; project administration, S.W.; funding acquisition, S.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by in part by the National Natural Science Foundation of China (grant no. 61773378, U1713222, and U1806204), in part by the Equipment Pre-Research Field Fund (grant no. 61403120407), in part by the Opening Project of Guangdong Provincial Key Lab of Robotics and Intelligent System.

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

## References

1. Liu, N.; Lu, T.; Cai, Y.; Wang, S. A Review of Robot Manipulation Skills Learning Methods. *Acta Autom. Sin.* **2019**, *45*, 458–470.
2. Bohg, J.; Morales, A.; Asfour, T.; Kragic, D. Data-Driven Grasp Synthesis: A Survey. *IEEE Trans. Robot.* **2014**, *30*, 289–309. [CrossRef]
3. Goldfeder, C.; Allen, P.K.; Lackner, C.; Pelossof, R. Grasp planning via decomposition trees. In Proceedings of the IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 4679–4684.
4. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep reinforcement learning: A brief survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
5. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998.

6. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiler, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
7. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of Go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef] [PubMed]
8. Kroemer, O.; Niekum, S.; Konidaris, G. A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms. *arXiv* **2019**, arXiv:1907.03146.
9. 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 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, BC, Canada, 24–28 September 2017; pp. 23–30.
10. Zhang, J.; Tai, L.; Yun, P.; Xiong, Y.; Liu, M.; Boedecker, J.; Burgard, W. Vr-goggles for robots: Real-to-sim domain adaptation for visual control. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1148–1155. [CrossRef]
11. Calinon, S.; Dhalluin, F.; Sauser, E.; Caldwell, D.; Billard, A. Learning and Reproduction of Gestures by Imitation. *IEEE Robot. Autom. Mag.* **2010**, *17*, 44–54. [CrossRef]
12. Zhang, T.; McCarthy, Z.; Jow, O.; Lee, D.; Chen, X.; Goldberg, K.; Abbeel, P. Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018; pp. 5628–5635.
13. Rahmatizadeh, R.; Abolghasemi, P.; Behal, A.; Boloni, L. From virtual demonstration to real-world manipulation using LSTM and MDN. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, 2–7 February 2018; pp. 6524–6531.
14. Codevilla, F.; Miiller, M.; Lopez, A.; Koltun, V.; Dosovitskiy, A. End-to-End Driving Via Conditional Imitation Learning. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018; pp. 4693–4700.
15. Ross, S.; Melik-Barkhudarov, N.; Shankar, K.S.; Wendel, A.; Dey, D.; Bagnell, J.A.; Hebert, M. Learning monocular reactive UAV control in cluttered natural environments. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 1765–1772.
16. Levine, S.; Pastor, P.; Krizhevsky, A.; Ibarz, J.; Quillen, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Rob. Res.* **2018**, *37*, 421–436. [CrossRef]
17. Pinto, L.; Gupta, A. Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 3406–3413.
18. Stulp, F.; Theodorou, E.A.; Schaal, S. Reinforcement Learning With Sequences of Motion Primitives for Robust Manipulation. *IEEE Trans. Robot.* **2012**, *28*, 1360–1370. [CrossRef]
19. Duguleana, M.; Barbuceanu, F.G.; Teirelbar, A.; Mogan, G. Obstacle avoidance of redundant manipulators using neural networks based reinforcement learning. *Robot. Comput. Integr. Manuf.* **2012**, *28*, 132–146. [CrossRef]
20. Althoefer, K.; Krekelberg, B.; Husmeier, D.; Seneviratne, L. Reinforcement learning in a rule-based navigator for robotic manipulators. *Neurocomputing* **2001**, *37*, 51–70. [CrossRef]
21. Miljkovic, Z.; Mitic, M.; Lazarevic, M.; Babic, B. Neural network reinforcement learning for visual control of robot manipulators. *Expert Syst. Appl.* **2013**, *40*, 1721–1736. [CrossRef]
22. Kakas, A.C.; Cohn, D.; Dasgupta, S.; Barto, A.G.; Carpenter, G.A.; Grossberg, S.; Autonomous Helicopter Flight Using Reinforcement Learning. In *Encyclopedia of Machine Learning*; Springer: Boston, MA, USA, 2011; pp. 53–61.
23. Kormushev, P.; Calinon, S.; Caldwell, D.G. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics* **2013**, 3, 122–148. [CrossRef]
24. Kober, J.; Bagnell, J.A.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Rob. Res.* **2013**, *32*, 1238–1274. [CrossRef]
25. Zhang, F.; Leitner, J.; Milford, M.; Upcroft, B.; Corke, P. Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control. In Proceedings of the Australasian Conference on Robotics and Automation, Canberra, Australia, 2–4 December 2015.

26. Zhang, F.; Leitner, J.; Milford, M.; Corke, P. Modular deep q networks for sim-to-real transfer of visuo-motor policies.In Proceedings of the Australasian Conference on Robotics and Automation, Sydney, Australia, 11–13 December 2017.
27. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 387–395.
28. Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.I.; Abbeel, P. Trust Region Policy Optimization. In Proceedings of the International Conference on Machine Learning, Lille, France, 6-11 July 2015; pp. 1889–1897.
29. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Harley, T.; Lillicrap, T.P.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
30. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv* **2015**, arXiv: 1506.02438.
31. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
32. Levine, S.; Finn, C.; Darrell, T.; Abbeel, P. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.* **2016**, *17*,1334–1373.
33. Stephen, J.; Edward, J. 3d simulation for robot arm control with deep q-learning. In NIPS 2016 Workshop: Deep Learning for Action and Interaction. *arXiv* **2016**, arXiv:1609.03759.
34. Mahler, J.; Liang, J.; Niyaz, S.; Laskey, M.; Doan, R.; Liu, X.; Ojea, J.A.; Goldberg, K. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv* **2017**, arXiv:1703.09312.
35. Viereck, U.; Pas, A.; Saenko, K.; Platt, R. Learning a visuomotor controller for real world robotic grasping using simulated depth images. *arXiv* **2017**, arXiv:1706.04652.
36. Fang, K.; Bai, Y.; Hinterstoisser, S.; Savarese, S.; Kalakrishnan, M. Multi-task domain adaptation for deep learning of instance grasping from simulation. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018; pp. 3516–3523.
37. Stein, G.J.; Roy, N. Genesis-rt: Generating synthetic images for training secondary real-world tasks. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018; pp. 7151–7158.
38. Cutler, M.; How, J.P. Efficient reinforcement learning for robots using informative simulated priors. In Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, USA, 26–30 May 2015; pp. 2605–2612.
39. Bousmalis, K.; Irpan, A.; Wohlhart, P.; Bai, Y.; Kelcey, M.; Kalakrishnan, M.; Downs, L.; Ibraz, J.; Pastor, P.; Konolige, K; et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018; pp. 4243–4250.
40. Rusu, A.A.; Vecerik, M.; Rothorl, T.; Heess, N.; Pascanu, R.; Hadsell, R. Sim-to-Real Robot Learning from Pixels with Progressive Nets. In Proceedings of the 1st Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017; pp. 262–270.
41. Sadeghi, F.; Levine, S. CAD2RL: Real Single-Image Flight Without a Single Real Image. In Proceedings of the Robotics: Science and Systems XIII; Robotics: Science and Systems Foundation, Cambridge, MA, USA, 12–16 July 2017.
42. Zhu, Y.; Wang, Z.; Merel, J.; Rusu, A.; Erez, T.; Cabi, S.; Tunyasuvunakool, S.; Kramar, J.; Hadsell, R.; de Feritas, N.; et al. Reinforcement and Imitation Learning for Diverse Visuomotor Skills. In Proceedings of the Robotics: Science and Systems XIV; Robotics: Science and Systems Foundation, PA, USA, 26–30 June 2018.
43. Peng, X.B.; Andrychowicz, M.; Zaremba, W.; Abbeel, P. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018; pp. 3803–3810.
44. Yan, M.; Frosio, I.; Tyree, S.; Kautz, J. Sim-to-Real Transfer of Accurate Grasping with Eye-In-Hand Observations and Continuous Control, Neural Information Processing Systems (NIPS) Workshop on Acting and Interacting in the Real World: Challenges in Robot Learning. *arXiv* **2017**, arXiv:1712.03303.
45. Zhang, F.; Leitner, J.; Milford, M.; Corke, P.I. Sim-to-real transfer of visuo-motor policies for reaching in clutter: Domain randomization and adaptation with modular networks. *arXiv* **2017**, arXiv: 1709.05746.

46. Shelhamer, E.; Long, J.; Darrell, T. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 640–651. [CrossRef]
47. Simonyan K.; Zisserman A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
48. Todorov, E.; Erez, T.; Tassa, Y. MuJoCo: A physics engine for model-based control. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Algarve, Portugal, 7–12 October 2012; pp. 5026–5033.
49. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
50. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.