Robotica (2024), **42**, pp. 1074–1093
doi:10.1017/S0263574724000092

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

One-shot sim-to-real transfer policy for robotic assembly via reinforcement learning with visual demonstration

Ruihong Xiao[1], Chenguang Yang[1], Yiming Jiang[2] and Hui Zhang[2]

[1]School of Automation Science and Engineering, South China University of Technology, Guangzhou, China and [2]The National Engineering Research Center for Robot Visual Perception and Control, Hunan University, Changsha, China
Corresponding author: Chenguang Yang; Email: cyang@ieee.org

Received: 9 March 2023; **Revised:** 10 December 2023; **Accepted:** 22 December 2023;
First published online: 24 January 2024

Keywords: visual demonstration; RL from demonstration; sim-to-real transfer; robotic assembly

Abstract

Reinforcement learning (RL) has been successfully applied to a wealth of robot manipulation tasks and continuous control problems. However, it is still limited to industrial applications and suffers from three major challenges: sample inefficiency, real data collection, and the gap between simulator and reality. In this paper, we focus on the practical application of RL for robot assembly in the real world. We apply enlightenment learning to improve the proximal policy optimization, an on-policy model-free actor-critic reinforcement learning algorithm, to train an agent in Cartesian space using the proprioceptive information. We introduce enlightenment learning incorporated via pretraining, which is beneficial to reduce the cost of policy training and improve the effectiveness of the policy. A human-like assembly trajectory is generated through a two-step method with segmenting objects by locations and iterative closest point for pretraining. We also design a sim-to-real controller to correct the error while transferring to reality. We set up the environment in the MuJoCo simulator and demonstrated the proposed method on the recently established The National Institute of Standards and Technology (NIST) gear assembly benchmark. The paper introduces a unique framework that enables a robot to learn assembly tasks efficiently using limited real-world samples by leveraging simulations and visual demonstrations. The comparative experiment results indicate that our approach surpasses other baseline methods in terms of training speed, success rate, and efficiency.

1. Introduction

Assembly task is a common basic task in robot manufacturing industry, including axle assembly, printed-circuit board assembly, and so on, which faces high density, high complexity, large scale, and flexible production needs [1]. Applying reinforcement learning (RL) to robot assembly tasks can help to meet the above requirements. In recent years, deep reinforcement learning (DRL) algorithms have fueled a wealth of publicized achievements of artificial intelligence in robotics [2–5]. However, there are still some limitations in terms of reliability and robustness. Indeed, some recent research shows that the existing DRL methods are brittle, brutal to transfer, unreliable in reality and sometimes over-fitting. It is still difficult to generate an ideal skill model for a robot manipulator [6]. Therefore, it is necessary to find a way to realize the practical application of robot RL.

Path planning is a traditional method to solve the robot manipulation problem, but sometimes difficult to meet with an unstructured environment and join the expert experience [7]. RL is a novel and effective method for path planning, which can help robots automatically learn about environmental changes to optimize models in an unknown environment. However, the exploration-exploitation dilemma is a motivating challenge for the performance of RL algorithms [8]. We introduce the enlightenment learning incorporated to balance exploitation and exploration, as excessive exploration leads to a decrease in cumulative returns, while excessive development locks agents in local optima. Learning from demonstration (LfD) is a natural solution for joining the expert experience [9]. References [10, 11] adequately

illustrate that human-inspired approaches can effectively improve the performance of robots. Imitation learning can use demonstrations of successful behavior to train policies that imitate the expert [12]. However, how to obtain the motion closer and better to the human experience is still a challenge. Reference [13] proposed a visual method to generate the demonstration data which is more human-like and can quickly train the robot for different manipulation strategies without otherwise complicated manual programs. However, vision-based demonstration introduces new problems. It needs to convert the human hand motion into robot hand motion to command the robot, called motion retargeting. In this paper, we adopt accurate camera calibration and coordinate transformation to solve that.

The pretraining process is pivotal in RL, offering advantages such as accelerated convergence, stability, and improved sample efficiency [14, 15]. It provides a beneficial starting point by initializing the model with task-specific knowledge, helping mitigate issues like poor convergence, and facilitating the transfer of learning between related tasks. In the realm of DRL, pretraining is particularly valuable for initializing neural network weights, overcoming challenges associated with deep network training, and guiding more informed exploration strategies. Overall, pretraining enhances the overall efficiency, robustness, and adaptability of RL algorithms, especially in complex environments with high-dimensional state spaces [15]. The success of pretraining in RL relies on the quality and size of the dataset used. A high-quality dataset should be representative, diverse and capture relevant aspects of the target environment, providing the model with a comprehensive understanding of the task. The dataset's size is a critical factor, as larger datasets offer more diverse experiences, enabling better generalization. However, the size should be balanced with computational considerations [16]. A dataset's relevance to the target task is paramount, ensuring that task-specific features contribute to the model's adaptability. Striking the right balance between dataset quality and size is essential for optimizing the pretraining process and enhancing the model's overall performance. In this paper, leveraging human visual demonstrations in pretraining datasets for RL enhances the model's adaptability, decision-making processes, and overall performance, while providing a rich, diverse, and contextually relevant set of experiences.

Owing to the limitation of gathering real-world data, that is, sample inefficiency and difficulty in data collection, simulation environments are more convenient for training RL agents. Nonetheless, when we prefer to transfer the agent trained by the simulation to the real world, the gap between reality and simulation will reduce the performance of the policy. Multiple research efforts toward closing the sim-to-real gap and solving the problem of sim-to-real transfer. As in survey [17], the main methods of sim-to-real transfer in DRL being utilized at the moment include these aspects: domain randomization, domain adaptation, imitation learning, meta-learning, and knowledge distillation. In order to complete the learning task of robot assembly skills for general use in reality, in this paper, we propose a novel framework that can transfer the policies trained in simulation to the real environment with a sim-to-real controller as shown in Fig. 1. The problems addressed in this paper are as follows:

How to generate more human-like demonstration in reality: We introduce a two-step method to develop manipulation trajectory from a single third-person-view visual demonstration. Our approach uses a binocular camera mounted on the outside of the robot. Then we transfer the trajectory from the camera coordinate system to the robot coordinate system through the calibration parameters of the camera.

How to train the RL model more efficiently: There are three main ways to incorporate demonstration experience in RL: pretrain the RL model, add the imitation learning loss while training, and put the demonstration data into a replay buffer. Since proximal policy optimization (PPO) is an on-policy RL method, we pretrain the RL network to learn the expert experiment.

How to fill up the sim-to-real gap: By collecting visual information with the binocular camera in reality, it is able to generate the demonstration trajectory and achieve automatic error correction to fill up the reality gap. In addition to this, we also adopt the domain randomization method during training to improve the robustness of the policy.

We carry out our experiments in the MuJoCo simulator. The primary contributions of this paper can be summarized as follows:

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

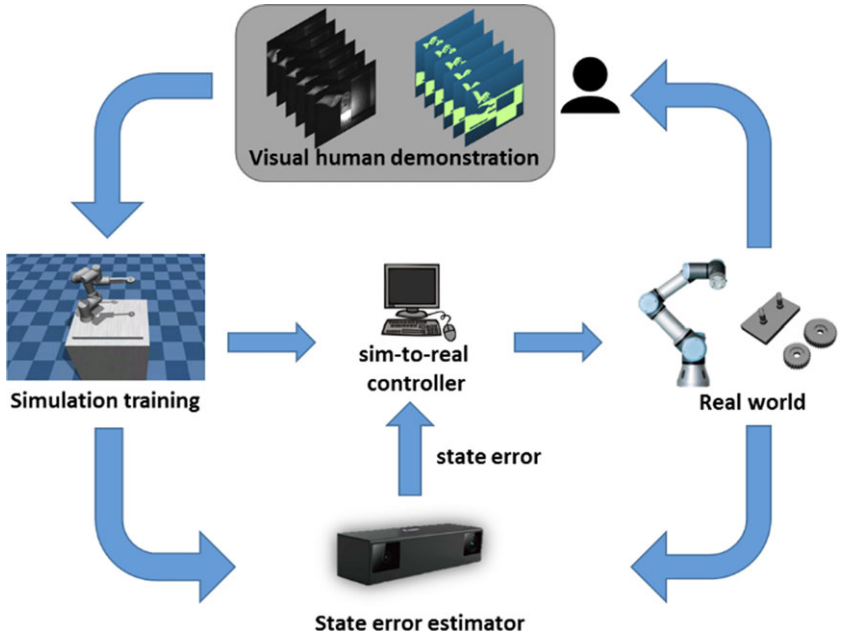

Figure 1. *An overview of our method: 1) gather human demonstration data by visual method, 2) robot executes the current policy in real world, 3) camera collect sim-to-real error, 4) simulation training of a PPO agent with demonstration, 5) a controller for sim-to-real transfer.*

1. We propose a vision-based method to extract demonstration trajectories in realistic industrial scenarios by SoLo and iterative closest point (ICP) to accurately extract realistic teaching trajectories within 2 mm of accuracy. We improve the ICP and make it more accurate.
2. We develop an assembled RL training environment and successfully transferred real-world data to simulation training. We assume the agent is in an unknown unstructured environment and learns by exploration. We used enlightenment learning to improve PPO to learn from the demonstration experience. The success rate increased from 80% to 96%, and the generated policy can achieve the target faster.
3. In order to apply the policies in realistic industrial application scenarios, we reduce the sim-to-real gap by a visual error estimation and develop a sim-to-real controller. We discuss the proposed approach on the recently established NIST gear assembly benchmark [18] to make a professional comparison. We compare the proposed method with the demonstration and some other similar approaches, such as original PPO and DDPGfD.

The rest of the paper is organized as follows. In the next section, we will introduce recent research on the related work. Section 3 introduces the preparatory knowledge required for this paper. Section 4 presents the overall framework and introduce each part of our system framework in detail. This section shows the contribution of the proposed approach. Section 5 shows the details of the experimental result. Finally, we draw up our conclusions in Section 6. Our studies show great performance of the proposed methods on efficiency.

2. Related work

Path planning is a key research issue in the field of robotic applications including local and global planning. For robot assembly, planning includes strategic planning and trajectory planning. Reference [19]

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

used the analysis of contact states in insertion stage and the forces conditions to make a strategy planning and combined with impedance control. Reference [20] develop a planning strategy of path finding and grasp planning. The trajectory planning methods include A* [21], D* [22], RRT [23], and etc. The map-based motion planning methods are numerically demonstrated, but when encountering unknown or unstructured environments, their effectiveness is not particularly good. Reference [24] summarizes the representative and state-of-the-art works of the classical motion planning architecture and RL-based approaches. As the complexity and randomness of the environment, the planning capability of the classical hierarchical motion planners is challenged. Reference [7] illustrated that most of the planning method for robotic assembly are found to be sensing information based and not on the integration of the sensing information and the environmental constraint. Human-inspired methods have not yet been applied to assembly tasks either. Reference [25] illustrated that DRL shows a better fitting performance with a tighter clearance and robustness against positional and angular errors for the peg-in-hole task. We establish the assembly problem in an unknown unstructured environment and try to obtain an effective and robust strategy in the paper.

Recent research shows possibilities to learn manipulation skills with RL. The existing RL algorithms applied in the field of robotics include TD3 [2], DDPG [3], PPO [4], SAC [5], and so on. It can be divided into model-free and model-based methods or into on-policy and off-policy methods. In this paper, we apply PPO, a model-free and on-policy RL algorithm, to realize the manipulation task of the robot. Model-free RL methods are capable of training an agent to act directly on the environment. The on-policy RL also helps the training converge faster. However, reference [24] introduced the challenge of RL-based motion planning methods, such as reality gap, reward sparsity problem, low sample efficiency, and generalization problem. To settle these problem, we have conducted research and found some effective methods.

LfD can be used to combine RL with demonstrations, such as dynamic movement primitives (DMPs) [26–29]. These methods use trajectory-centric policy representations that are well suited for imitation but do not enable feedback on rich sensory inputs. In some recent work, demonstrations have been used for pretraining the policy, such as DQfD [30]. Pretraining RL policies using trajectories from an expert can help to accelerate training process. On the other hand, behavior cloning (BC) treats the problem of imitation learning, i.e., using expert demonstrations, as a supervised learning problem [31], which is also an effective imitation learning method. DDPGfD [15] successfully combines RL with demonstrations by incorporating the demonstrations into the RL algorithm. It is suited for off-policy RL method by adding them to the replay buffer. Besides, DAPG [14] bootstraps the policy using behavior cloning and combines demonstrations with an on-policy policy gradient method. On-policy RL methods, on the other hand, are more stable and scale well to high-dimensional spaces.

The traditional demonstrating method needs to drag the end of the robot to perform the manipulation task or act with the teleoperation [32, 33]. Instead, by acquiring the demonstration data in a visual way, the expert experience obtained is closer to human actions. Such as [13], the author proposes a closed-loop, category-level manipulation framework based exclusively on visual feedback with a robot teaching method from a single visual demonstration. For the visual demonstrating method, pose estimation is an important part. In some recent research, there are also some end-to-end pose estimation methods based on deep learning [34, 35], but the precision and accuracy of these methods are difficult to achieve the desired results. With the development of computer vision, instance segmentation has been able to recognize and segment targets effectively, such as Mask-RCNN [36] and SoLo [37]. For pose estimation, leverage geometric estimators such as RANSAC/PnP [38] and feature matching such as PFH [39] and FPFH [40] can be used to solve object poses, but the processing time is relatively long. In our work, we proposed a two-step object pose estimation method using instance segmentation [36, 37] and ICP [41] to extract the pose from the visual data and produce a demonstrative trajectory. Benefiting from the geometric information contained in the point cloud, the pose estimation method of combining the image and point cloud is more accurate.

In recent years, there have been some transfer methods for RL agents to act in the real world. References [42] use a method similar to digital twins with force information for zero-shot sim-to-real

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

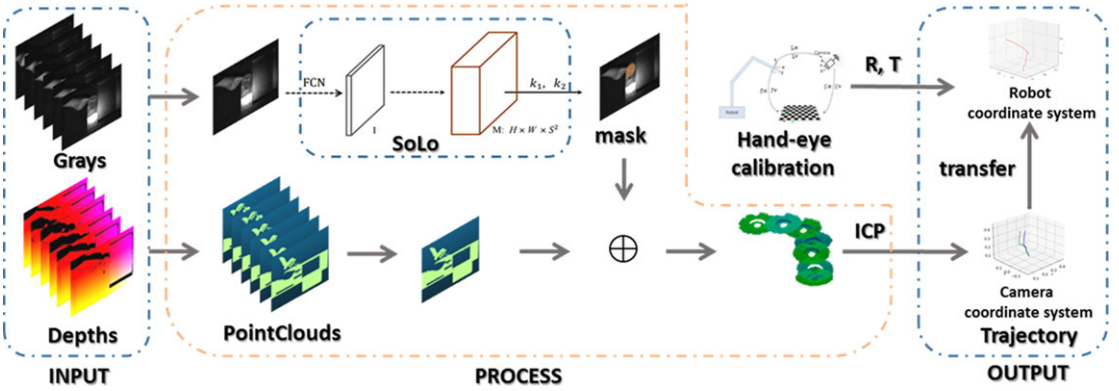

Figure 2. *The process to extract trajectory from the visual human demonstration.*

transfer. It corrects the errors in real time through the force sensor and achieves a lower reality gap. Reference [43] uses domain randomization for transfer and help raise in domain error, but substantially reducing out of domain error. Domain randomization also plays an important role in error correction in our work. Reference [44] uses the domain adaption, which crosses the sim-to-real gap by mapping the simulation and reality into a latent space. On the other hand, using real-world data also benefits transferring the model. In this paper, we use a visual demonstration approach that allows direct access to human expert experience.

3. Preliminary

Figure 2 shows the visual demonstration trajectory extraction method proposed in this article. To generalize a trajectory from the visual demonstration, we need to first identify the object and estimate its position. Thus, we proposed a two-step object pose estimation method using instance segmentation and ICP. In this paper, we can achieve a fast identification and segmentation of the target by SoLo. Then we can generate the point cloud of the object from the mask. We used improved ICP [41] algorithm of the point cloud to achieve the estimation of the target pose. In the experiments of this paper, about 20 iterations of each point cloud are performed to obtain the accurate target position pose.

The RL problem can be defined as a policy search in a Markov decision process (MDP). RL can be divided into two types: on-policy and off-policy RL. The PPO algorithm uses fixed-length trajectory segments. In each iteration, we collect T timesteps of data and store each transition (s, a, r, s') in memory to calculate the advantages. (s, a, r, s') represents the state, action, reward, and the next state. $V(s)$ is the value of s calculated by the critic. The advantage-function estimator is defined as follows:

$$\begin{aligned} \hat{A}_t &= \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \\ \delta_t &= r_t + \gamma V(s_{t+1}) - V(s_t) \end{aligned} \tag{1}$$

where $\hat{A}_t$ specifies the advantage at t in [0, T], within a given length-T trajectory segment. γ is the discount factor of the reward, and λ is the discount factor of the steps. After calculating the advantages, we should set a hyperparameter ϵ for the cipped surrogate objective. The clipped surrogate objective is calculated as follow:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{2}$$

To update the neural network architecture that shares parameters between the policy and value function, the loss function consists of three terms: the policy surrogate, a value function error, and an entropy bonus.

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

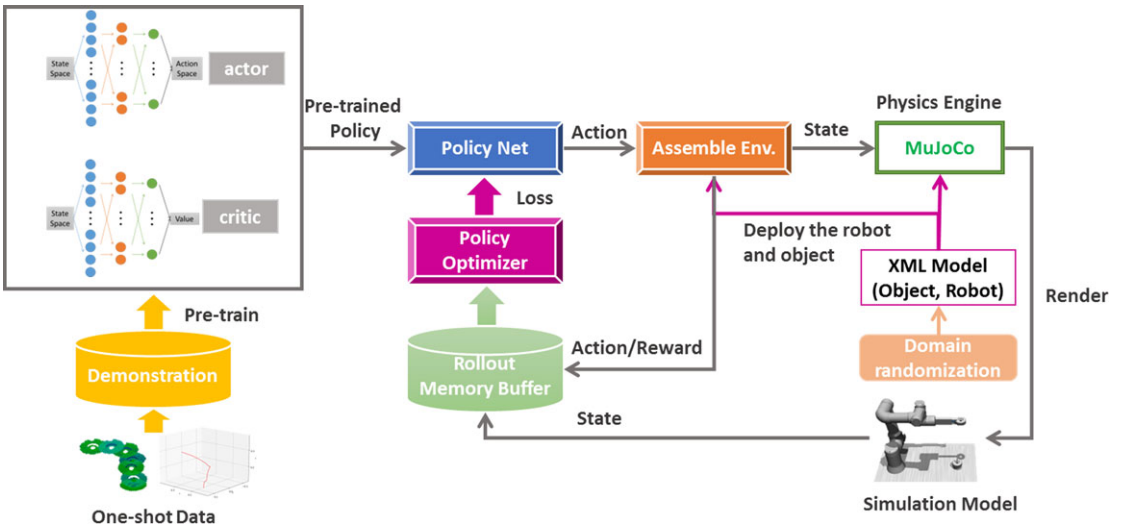

Figure 3. *Training pipeline: the single visual data is add into the demonstration pool to pretrain the actor and critic of the policy net. And then the policy net is trained in the MuJoCo with PPO algorithm.*

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S\left[\pi_\theta\right](s_t) \right] \tag{3}$$

where c_1, c_2 are coefficients, S denotes an entropy bonus, and L_t^{VF} is a squared-error loss $\left(V_\theta(s_t) - V_t^{\text{targ}}\right)^2$.

In reference [4], there are two approaches to complete the PPO algorithm, including the clipped surrogate objective and the adaptive KL penalty coefficient. In this paper, we only used the former approach because of the weak effect of the latter method on the results.

4. Methods

In this paper, we propose a training pipeline as shown in Fig. 3 for robots to learn assembly skills. The system framework consists of three major parts: demonstration pretraining, PPO training, and sim-to-real transferring. To work in the real world, we estimate the sim-to-real error through the vision and design a sim-to-real controller to fill up the sim-to-real gap. For transferring the training model from sim to real, we also adopt domain randomization while training which plays an important role in error correction. In our framework, all the demonstration steps are completed in reality.

4.1. Visual demonstration

For the high-precision manipulation task with rich contact proposed in this article, the agent needs to perform ordered fine-grained motion to achieve success. Adding demonstration is an intuitive way to display possible solutions to tasks to agents and can guide reasonable initial strategies. Due to the lack of clear modeling potential factors, expert demonstrations provided by humans typically exhibit significant variability [45]. We proposed a novel method for extracting high-precision visual teaching trajectories. For the demonstration video frame, the object state is extracted via a two-step pose estimation method, which allows to represent of task demonstration with an extracted trajectory. Some end-to-end pose estimation methods [34, 35] are able to do so but with unacceptable accuracy. We improve the accuracy of visual demonstration trajectory through point cloud processing and optimization methods. First, we collect a set of artificial teaching data of the image and depth information by a binocular camera with structured light. We train a SoLo model for object recognition and segmentation, which can achieve fast instance segmentation. We get deep information about the mask of the target object. By joining

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

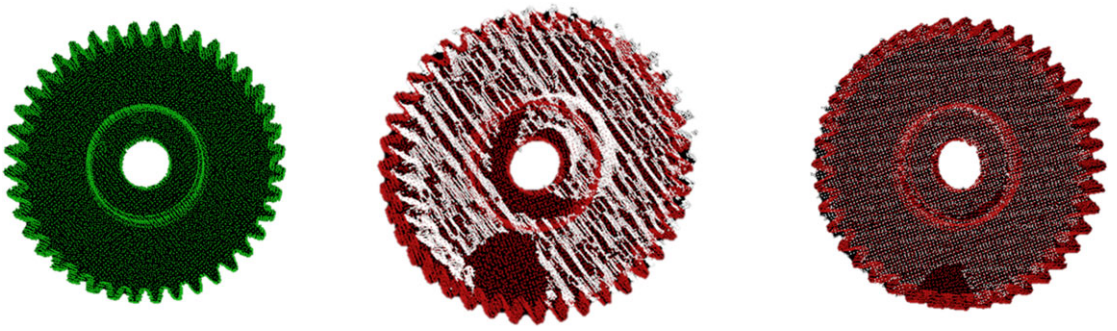

Figure 4. *The improved ICP method. The green and red point cloud is the original point cloud of the gear. The second image is the original ICP registration result. The third image is the improved ICP point cloud registration result. The white point cloud is the input point cloud, and the red point cloud is the expected point cloud position.*

the depth information, we can use the internal parameters of the camera to obtain the real point cloud. We preprocess the point cloud, including downsampling, radius filtering, and Euclidean clustering to improve the ICP as shown in Fig. 4. Compared to the end-to-end method, the 6D pose obtained by this method is closer to the actual state and easier for error analysis. The error of the position estimation method we use is about 2 mm.

In this work, we have used the OpenCV and PCL libraries to process the image and point cloud data. After generating the teaching trajectory, we need to convert the teaching trajectory from the image coordinate system to the world coordinate system by hand-eye calibration. We convert the pixel coordinates of the target to the world coordinate system through the internal and external parameters of the camera. The equation for the transformation is as follows:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ \overrightarrow{0} & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{5}$$

where u, v is the pixel position in the image. (u_0, v_0) is the pixel position in the center of the image. dx and dy denote how long each column and row represents. f is the focal length of the camera, which we often refer to as the internal parameter. (x_c, y_c, z_c) represents the position in the camera framework. (x_w, y_w, z_w) is the real position in the world. $\boldsymbol{R}$ is the rotation matrix, and $\boldsymbol{T}$ is the transfer vector between the camera coordinate system and the world coordinate system, which we often refer to as the external parameter. In the real world, we can extract the internal and external parameters of the camera by hand-eye calibration.

4.2. Reinforcement learning

There are a number of simulators developed in response to the rapid development of RL in robotics, such as Pybullet [46], MuJoCo [47], and Isaac. We consider the requirements to meet the demands of our experiment. MuJoCo is chosen according to the comparison of different simulators [48]. We also develop RL algorithms by using OpenAI Gym, which is utilized as a convenient toolkit for RL algorithms.

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

PPO has been illustrated to work in the field of continuous spatial control of robots, which is one of the commonly used on-policy RL methods for robot skill learning. Compared with the off-policy method such as DDPG, PPO has better stability and adaptability. We improve the PPO algorithm, and there are three modifications that significantly increase performance of the original PPO algorithm: 1) The experience data retains the demonstration and exploration data, and actions from the human are regarded as the agent's own. 2) We have designed a simple and efficient training environment and a new shaped-reward function for the robot gear assembly task. 3) We added noise to the parameters and the actor's output – thus we have a robust policy.

4.2.1. Architecture

We use an Actor-Critic style PPO algorithm with a policy network and a value network. Both of the neural networks have three linear hidden layers with 256, 128, and 64 units, respectively. The ReLU activation function is used in all hidden layers. The input state is three dimensional, the output action is three dimensional, and the output value is one dimensional. With a constraint of the clipped surrogate objective in the PPO algorithm, there would not be an excessively large policy update while training which helps to reduce policy forgetting. Due to this advantage of the PPO algorithm, the retrained network can better retain the experience from the pretraining model and generate a more general policy.

However, the shortcomings of traditional robot RL methods make it difficult for us to obtain a good assembly policy through RL directly, which affects its effectiveness and practicality in real-world applications. We introduce enlightenment learning incorporated via pretraining to overcome these shortcomings, which involves a combination of algorithmic advancements, improving data efficiency, balancing exploration and exploitation, accelerating training, and engineering rewards. Due to the fact that RL is a random exploration process, without pretraining the model, the trajectories and policy generated are unpredictable. The proposed training strategy can enable RL to obtain basic policy units in advance.

After transforming the demonstration data to the robot base coordinate system in the simulator, we can pretrain the policy network and the value network. First, all the demonstration data are transfered into (s, a, r, s') and put into the experience pool. While training, 64 groups of data are randomly extracted from the experience pool in each training epoch. After training for 2000 epochs, we can obtain a pretrained policy network and value network. The pretrained policy network and value network are the same as the networks for RL, including an actor and a critic. In this way, we can initialize the RL model for further training. After pretrain the RL model, PPO will control the rest of the training.

4.2.2. Environment

There is a gap in applying RL to real-world robot tasks. For example, in real-world scenarios, unexpected actions may lead to potential safety issues for robots, and low sampling efficiency in the real world may lead to convergence difficulties in the training process. It is necessary to create a training environment that can achieve the transfer of simulation and reality. We build a new simulation environment that can maintain transfer-ability between domains and train the agent in the Cartesian space. Simulation results illustrate that it speeds up learning process and reduces training time greatly. We train an agent in Cartesian space using proprioceptive information. The training of Cartesian space skills can greatly simplify the model and reduce the amount of data required for training, which is beneficial to transfer to the real world. The state space and the action space are defined as follows:

$$\begin{aligned} \boldsymbol{S_t} &= [x_t, y_t, z_t] \\ \boldsymbol{A_t} &= [\delta x, \delta y, \delta z] \end{aligned} \tag{6}$$

where (x_t, y_t, z_t) is the position of the end effector at t. $(\delta x, \delta y, \delta z)$ is the displacement of the end effector. We also normalize the state space and action space. In this paper, we trained an assembly policy with PPO to obtain the action of the end effector based on the current state of the environment. The actions

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

Table I. *Environmental constraint.*

Parameters (unit)	**Threshold**
Joints limit j_i (rad)	$[-3.14, 3.14]$
The X-axis coordinates of the end effector x_t (m)	$[0, 0.6]$
The Y-axis coordinates of the end effector y_t (m)	$[-0.5, 0.5]$
The Z-axis coordinates of the end effector z_t (m)	$[0, 0.5]$
Steps limit	64

generated by the policy are in the Cartesian space. Therefore, we need to transform the actions from Cartesian space into joint space. We design a solver to settle the inverse kinematics problem of robots with PyKDL library and control the robot in joint space.

Considering the safety during robot assembly and also for more convenient training, we also define a mechanism for environment reset. The environment is reset when: 1) The robot joint angles out of limit; 2) the end effector is out of the safe workspace; and 3) the number of steps exceeds the set threshold. The threshold is as shown in Table I.

4.2.3. Shaped reward

The RL method is guided by reward signals. Instead of using a sparse reward, we well design a shaped reward function, which is beneficial for improving sample validity and faster convergence. We use sparse rewards to guide exploration while using dense rewards to provide more timely feedback. The sparse reward is defined as follows:

$$\begin{cases} r_{done} = 0 & done = true \\ r_{done} = 1 & done = false \end{cases} \tag{7}$$

Sparse rewards make gradient propagation difficult, so it is necessary to design a dense reward function. Using a shaped reward function requires a large amount of engineering effort to extract the necessary state information. We compared different reward functions, including linear function, exponential function, gaussian function, and so on. These functions are simple but difficult to adapt to assembly tasks. We find that using a logarithmic function allows the environment to generate larger rewards when the target is closer. The reward function of the distance is defined as follows:

$$\begin{aligned} r_t &= d_t - log(d_t^2 + \epsilon) + \upsilon \\ d_t &= -\sqrt{(x_g - x_s)^2 + (y_g - y_s)^2 + (z_g - z_s)^2} \end{aligned} \tag{8}$$

where (x_g, y_g, z_g) is the position of the gear and (x_s, y_s, z_s) is the position of the shaft. d_t represents the distance between the current position and the target position at t. r_t is the reward at t. ϵ is a constant used to prevent the case where the logarithmic function of the reward fails to take a value for a distance of zeros, ϵ is set to 0.005. υ is a bias constant of the reward, υ is set to 5.0. The environment is reset when $d_t > 0.5$ or when the joint angle of the robot is greater than the joint limit.

Besides, we define a successful attempt as the robot completes the assemble task at least 0.02 m. When the task is completed, the environment generates a reward $r_{done} = 1$. In conclusion, the total reward is as follows:

$$\begin{cases} r_{total} = r_t & d_t > 0.02 \\ r_{total} = r_t + r_{done} & d_t <= 0.02, r_{done} = 1 \end{cases} \tag{9}$$

We collect a set of demonstration data mentioned in Section 4.1 and store each transition (s, a, r, s') to pretrain the RL model. In our framework, all the demonstration steps are completed in reality. After

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

Table II. *Domain randomization configuration.*

Parameters (unit)	μ	ρ	**Noise distribution**
Initial joint angles j_i (rad)	0	0.1	Uniform
Actions a_t (m)	0	0.01	Gaussian

pretraining the RL model, PPO will control the rest of the training. Finally, we obtain a usable model to complete the assemble task.

4.3. Sim-to-real transfer

In order not to overly adapt to any specific absolute coordinate position, we introduced a mechanism that allows agents to extend to new positions and fill up the sim-to-real gap. Directly using policy in the real world will consistently hurt performance. In our regime, we have found that domain randomization and sim-to-real error correction improved performance of transference. We performed experiments in reality to demonstrate the ability of our training model to transfer from simulation to reality. We use two methods to do this: training in a randomized environment and filling up the sim-to-real gap with a sim-to-real controller. The results show the trained model robust enough to work in different domains.

4.3.1. Domain randomization

While training, we added noise to the parameters and the ground truth position. In our framework, domain randomization works well in error correction. We used two types of randomization: observation randomization and dynamic randomization. The observation randomization represents the uncertainty of the target and the initial state. Dynamic randomization represents the model's inaccuracy while interacting with the environment. We used two types of noise distribution: Gaussian distribution $N_g(\mu, \rho)$ (where μ is the mean and ρ is the covariance) and Uniform distribution $N_u(\mu, \rho)$ (where μ is the mean and ρ is the absolute value of the upper and lower limits). The domain randomization configuration of the experiment is given in Table II.

4.3.2. Sim-to-real controller

It is necessary to create a control method that can achieve the transfer of simulation and reality. In this article, we successfully addressed this issue by designing an error estimator. The grasp planning algorithm used in this paper is the one proposed in reference [49]. Due to the error of the grasping position and object differences, there are differences between reality and simulation. In this paper, we assume that in the assemble task, the reality gap is caused by the different positions of the gear and shaft in the real world and simulation. So we design a sim-to-real controller to fill up the sim-to-real gap. We start by moving the robot in reality and simulation to the same initial state and then identify and calculate the state of the target in reality to calculate the error between reality and simulation. Before the policy starts, we have an initial estimate of the positions of the gears and shaft both in the real world and simulator. When the policy is executed in reality, we superimpose the acquired error on the policy output to control the robot in reality. The error is calculated as follows:

$$\boldsymbol{\Delta p} = \boldsymbol{p}_{real} - \boldsymbol{p}_{sim} \tag{10}$$

where $\boldsymbol{p}_{real}, \boldsymbol{p}_{sim}$ is the position of the assembly target in the initial state generated by the approach to estimate the 6D pose of the target mentioned in Section 4.1. $\boldsymbol{\Delta p}$ represents the sim-to-real error. While performing tasks, the policy is running in the simulation, and the real robot is controlled by the state of the robot in the simulation combined with error.

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

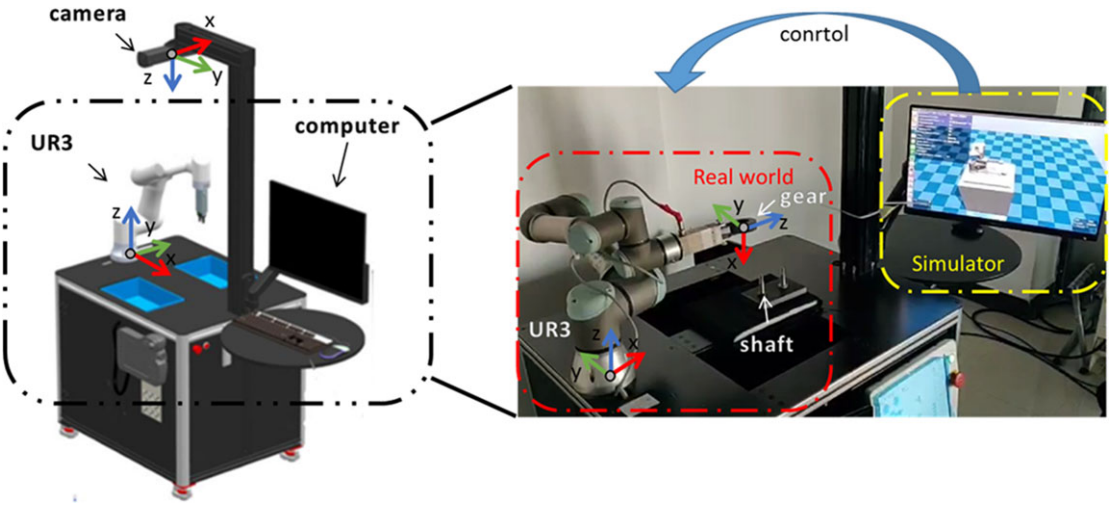

Figure 5. *The assemble setup with a UR3 robot in the real world and simulation.*

$$\begin{aligned} \Delta \boldsymbol{q} &= \boldsymbol{J}(\boldsymbol{q}) * \Delta \boldsymbol{p} \\ \boldsymbol{q}' &= \boldsymbol{q} + \Delta \boldsymbol{q} \end{aligned} \tag{11}$$

where $\boldsymbol{q}$ is the joint angles and $\boldsymbol{J(q)}$ is the Jacobian matrix of the robot at the current moment in the simulator. $\boldsymbol{q}'$ is the real joint angles to control the real robot.

5. Experiments

There are two different sizes of gears used in the experiments in the NIST gear assembly benchmark. The proposed method is utilized to perform manipulation of different gear objects with no additional programing. In the experiments presented in this paper, the following questions are explored: 1) What is the effect of the expert experience added through the visual demonstration? 2) How robust are the skills to different scenes and uncertainty due to dynamics? 3) Does our framework work well in real-world tasks and robots?

5.1. Simulator environment

We set up the environment in the MuJoCo simulator, which can effectively reduce the reality gap and demonstrate the proposed method in the gear shaft assembly task. The experimental environment consists of three main parts: simulator, robot, and assembled objects. Figure 5 shows the entire system in the real world.

5.1.1. Simulator

MuJoCo is a general-purpose physics engine that has achieved good results and applications in robotics. It is able to quickly and accurately simulate the interaction between articulated structures and the environment. Multiple cameras are set in the simulation environment to obtain the grasping position and teaching trajectory, including the camera on the robot hand and the camera fixed to the rear of the robot.

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

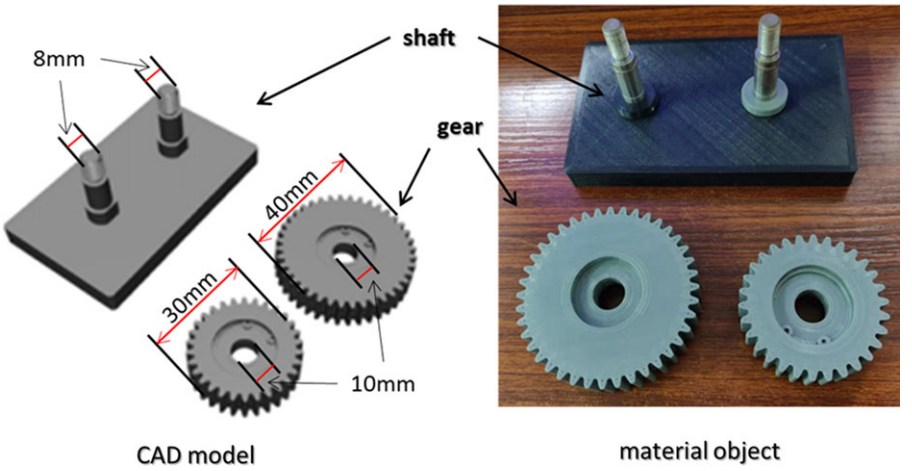

Figure 6. *The gears and shafts of the assembly task. The left picture shows the CAD model, and the right picture is the real object.*

5.1.2. Robot

We use UR3, a 6-DOF robot with a two-fingered gripper. In the simulator, we implement the joint control of the robot with the designed PD controller. Besides, controllers transform the high-level actions into low-level virtual motor commands that actuate the robots with the inverse kinematics of the robot. The URDF file of the robot is used to import the joint information of the robot, and the required robot end position is obtained to calculate the IK solution of the robot.

5.1.3. Assemble objects

As shown in Fig. 6, the gear assembly task consists of two main parts: the gear and the shaft. In this paper, two different gears are used, each mounted on a different shaft. The outer diameters of the gears are 40 and 30 mm, respectively, and both inner diameters are 10 mm. The diameters of the shafts are 8 mm.

5.2. Visual human demonstration

In this experiment, the COMATRIX camera was used to generate grayscale images, depth images, and pointclouds of the assemble target at the camera's acquisition rate (1–2 Hz). We have collected 120 grayscale images and depth images. The data we collect from the camera of the human demonstration are as follows:

1) Grayscale image: 2448*2048 pixels (as shown in Fig. 7, the first row)
2) Depth image: 2448*2048 pixels (as shown in Fig. 7, the second row)
3) Pointcloud: 1,628,584 points (as shown in Fig. 7, the third row)

After processing the raw data of teaching, we can obtain the trajectory of teaching actions in the camera coordinate system. With hand-eye calibration, the transformation matrix and translation vectors from the camera to the robot base coordinate frame can be obtained so that we can map the trajectory to the robot base coordinate system. The trajectory extraction method proposed in this paper has high accuracy and can obtain trajectories with human experience. It is with an error of 2 mm, which meets the requirements of the gear assembly task. Then we control the robot to perform the extracted trajectory and compare with the trajectory in the video. As shown in Fig. 8, the robot can accurately reproduce

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

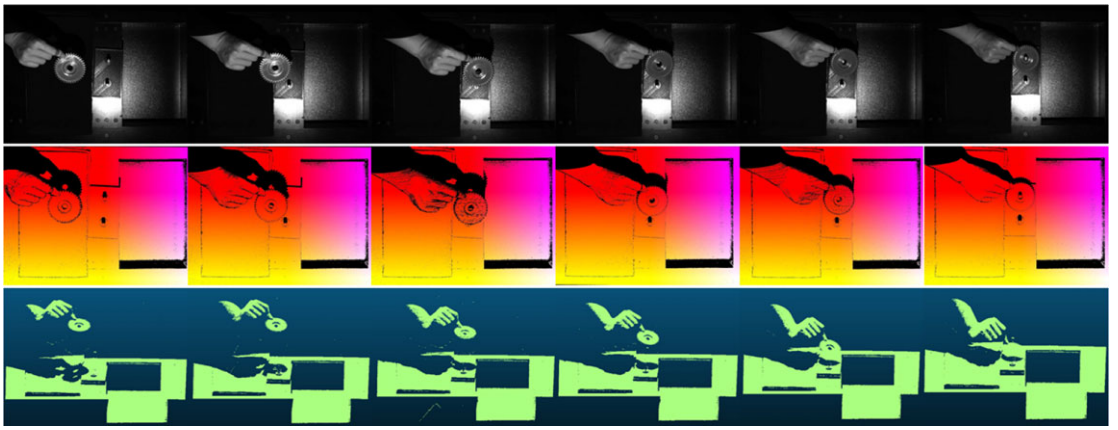

Figure 7. *The visual demonstration data. The first row is the gray scale images, the second row is the pseudo-color depth images, and the third row is the generated point clouds.*

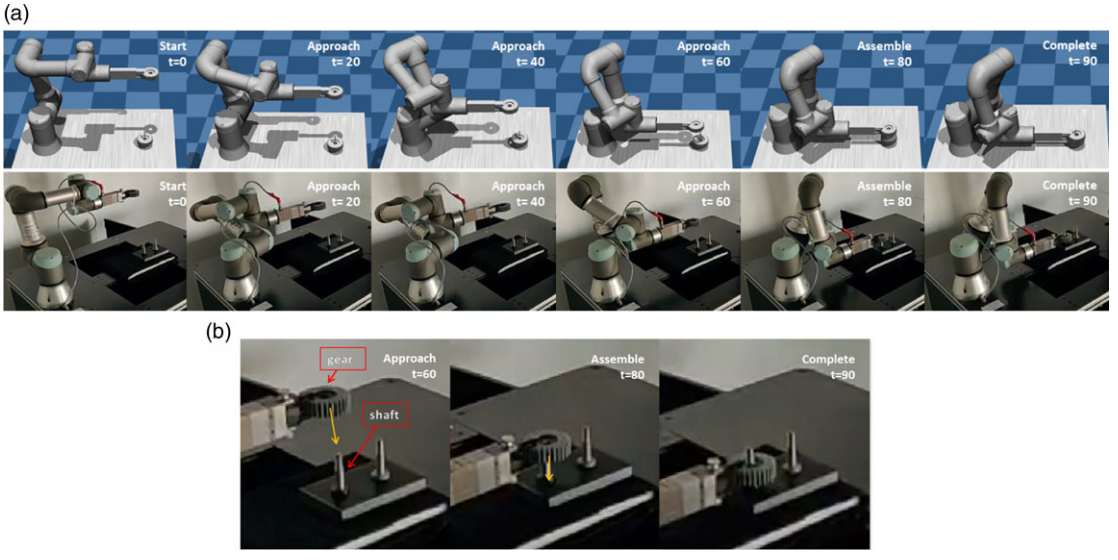

Figure 8. *The snapshots of the trained agent performed during the assembly task in both the simulator and the real world. (a) The first row is in the simulator, and the second row is in the real world. (b) The close-up of the gear assembly process.*

the action trajectory and successfully complete assembly tasks taught by humans, which proves that our method can accurately extract trajectories with human experience. The demonstration trajectory also has a good average reward result.

5.3. Performance of reinforcement learning

Trajectory planning for robots using RL involves training a policy that guides the robot's actions over time. In this context, the trajectory represents a sequence of states and actions that the robot takes to accomplish a task. The policy maps states to actions, effectively determining the robot's behavior. During the execution phase, the learned policy is applied to generate trajectories in real-world scenarios. The robot leverages its acquired knowledge to navigate and adapt to the environment, making decisions at each step based on the learned policy. This approach allows the robot to plan and execute

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

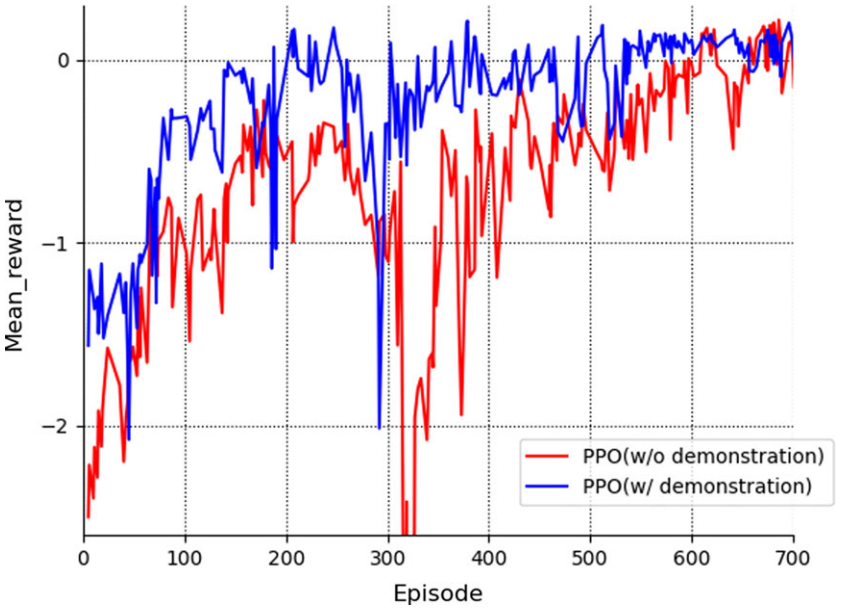

Figure 9. *The assemble task training reward curve under different training policies of PPO. The red curve is without the use of demonstration. The blue curve is obtained by adding the pretraining of the demonstration. If the policy can achieve the assembly target, the average reward will converge to around 0.*

trajectories that align with its learned understanding of optimal actions in different states, showcasing the adaptability and intelligence achieved through RL.

5.3.1. Training results

We test the generalization ability of our framework under different conditions and show the smoothed episodic reward with the smooth function is $r_{t+1}{}' = 0.4r_t' + 0.6r_{t+1}$. The training mean reward curve results of the PPO agent are shown in Fig. 9. We trained the policy with 700 episodes and each lasting 64 steps under different conditions. It is clear that the agent converges to a policy that allows it to successfully complete the task after nearly 600 episodes without demonstration and domain randomization. The mean reward per episode of the pretrained policy is about -1.3 at the start, and after re-training, the mean reward will reach the top after nearly 200 episodes. The reward and average reward in both modes start at a lower value and rise to a stable value. This indicates that the robot has acquired environmental knowledge through training and is able to adapt to the environment.

To test the generalization capability of the learned policies, we ask the agent to perform gear assembly tasks at new locations using the previously trained policies without any fine-tuning. The result shows that the model obtained by training again can well retain the experience obtained by pretraining and optimize the policy obtained by pretraining. As shown in Fig. 10, the policy after pretraining outperforms the policy without demonstration or trained with demonstration only. The policy with demonstration allows for stable execution of assembly actions along the shaft. Without the demonstration, the trained policy may go closer to the target position from the side of the shaft, which will lead to the failure of the assembly task.

5.3.2. Comparison experiment

We conducted contrast experiments comparing our method with the original methods. As shown in Table III, the proposed method has the least training time, the highest success rate, and the least number

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

Table III. *Table of the results of different trained policies.*

Policy	Training time	Average reward	Success rate	Average steps
Demo	–	−0.27	–	120
Pretrain	–	−0.66	96/100	134
PPO w/o Pretrain	600	−0.76	80/100	158
PPO w Pretrain	**200**	**−0.12**	**97/100**	**90**
DDPGfD	1300	−0.58	47/100	140

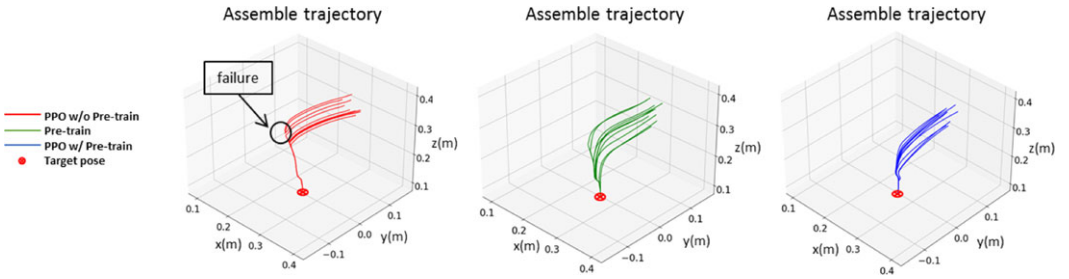

Figure 10. *The assembly trajectories result under different training policies of PPO. The red trajectories are without the use of demonstration data. The green trajectories are obtained directly by pretraining with the demonstration. The blue trajectories are obtained by adding the pretraining of the demonstration and retrained in simulator. The failure cases in the first figure are caused by the singular position of the robot.*

of execution steps. We selected three models trained by different methods: the model trained by the original PPO algorithm, the pretrained skill model, and the model obtained by training again after pretraining. In the simulation, one hundred assembly tasks are carried out for each model in the same environment. The initial state of each assembly task is randomized. If the assembly strategy can be completed within 200 steps, the assembly is judged to be successful. The success rate of assembly and the average number of steps are calculated and repeated ten times. The result is shown in Fig. 11. The average success rate and average number of step knots for the three models tested are, respectively, 80% and 158 steps, 96% and 134 steps, and 97% and 90 steps. Policy trained using the proposed method has a significantly higher success rate and can use fewer steps to complete the assembly task.

We tested the policies obtained from different PPO training strategies and extracted 10 different trajectories for each strategy. We randomized the initial position of the assembly and set the target position to be the same (0.25 m, 0.0 m, and 0.06 m). The assembly trajectories are shown in Fig. 10, and the position changes in three directions results are shown in Fig. 12. The assembly policy trained using the method proposed in this paper generates shorter and smoother motion trajectories. As shown in Fig. 12, the original PPO training method is unable to successfully complete assembly tasks at some random initial positions. Two out of the 10 samples were unable to successfully reach the target position to complete the assembly action. After adding pretraining with demonstration data, the success rate and assembly speed of the training policy are significantly improved.

In addition to comparing with the original PPO algorithm, we also compared with another similar RL method, DDPGfD which is an off-policy method. The trajectory generated by PPO production is smoother than that generated by DDPG due to the design method of PPO algorithm loss function. The trajectory generated by DDPGfD may experience some jitter. As shown in Fig. 13, we compare and discuss the generated trajectory of the proposed method with the demonstration and DDPGfD in three directions: x-axis, y-axis, and z-axis. All generated policies perform the same task. The starting position of the assembly is approximately (0.3, 0.12, 0.3), and the target position is (0.25 m, 0.0 m, 0.06 m). The

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

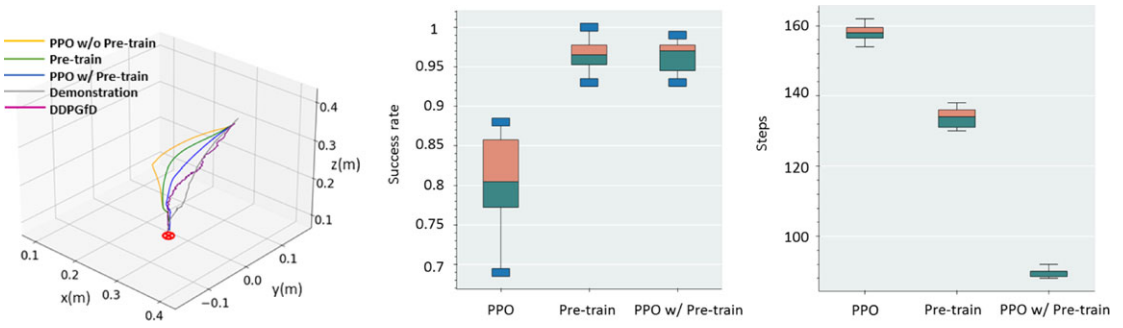

Figure 11. *The assembly trajectory, the average success rate and the average number of steps of different methods.*

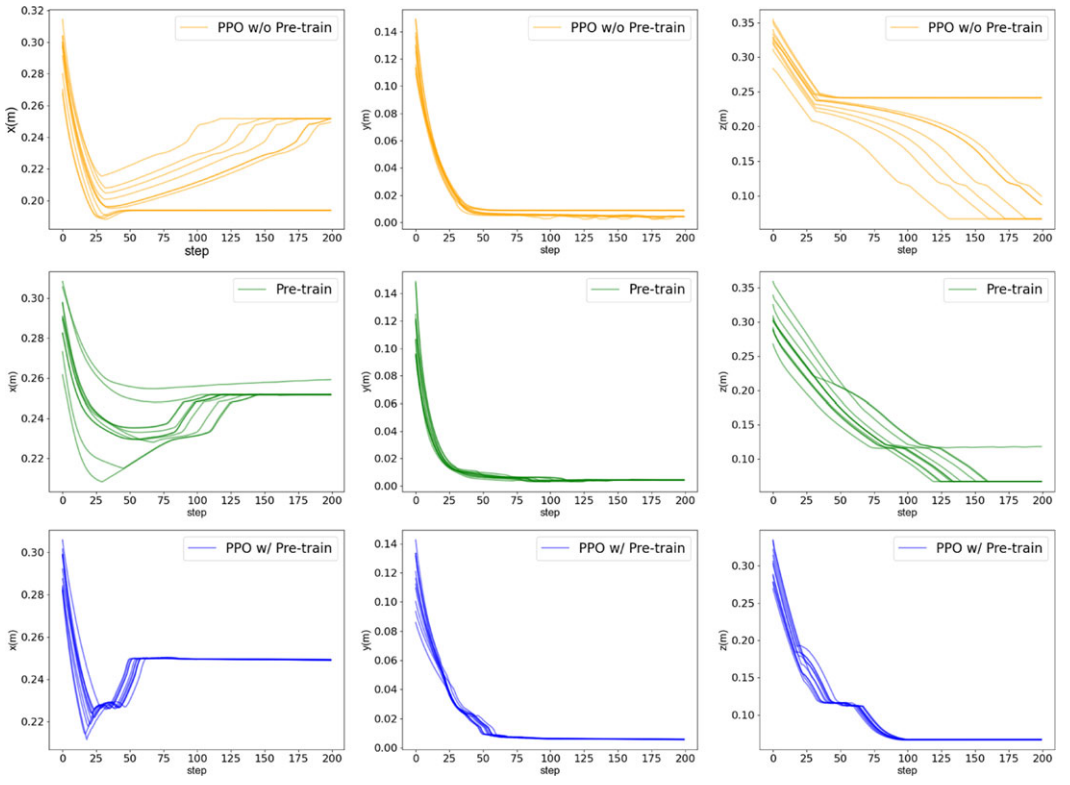

Figure 12. *Position changes in three direction of the original PPO algorithm without pretraining, the pretrained policy and the improved PPO algorithm with pretraining.*

results showed that the strategy obtained through PPO and pretraining reached the target position the fastest.

We also compared the changes in rewards during the assembly process in Fig. 14. Obviously, our method enables faster assembly. It outperforms vendor solutions by large margins in terms of perturbation ranges while keeping high success rates. In total, we have performed 100 trials on each method with a random initial position. This strongly implies our method is robust and reliable. Most failures occur due to active hard contact can cause changing dynamics. In the future, we plan to look at ways to encourage a more gentle insertion policy.

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

Table IV. *Table of the results of different sim-to-real transfer methods.*

Transfer methods	Success rate
w/o error estimator	15/20
w error estimator	18/20

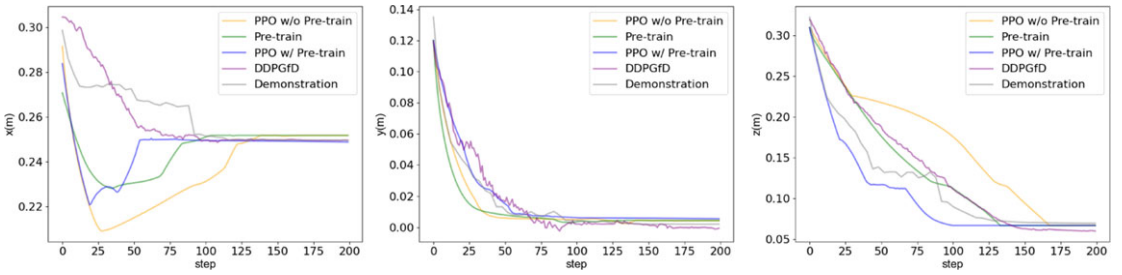

Figure 13. *Comparison of the demonstration trajectory, DDPGfD, and the experimental trajectory of different training policies.*

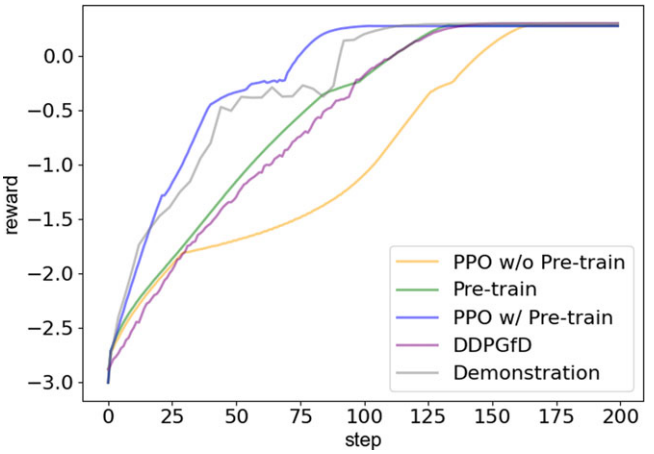

Figure 14. *The experimental reward comparison chart of different training methods, DDPGfD, and demonstration during the assembly process. When the assembly task is completed, the reward is close to 0.*

5.4. Sim-to-real transfer

In gear assembly tasks, the main sim-to-real gap comes from the position error of the assembly target and the motion error of the robot. In this paper, we develop an error estimator to settle the sim-to-real gap. Five groups of 20 repeated experiments with and without error estimation were done. The results are shown in Table IV. With the error estimator, the success rate of sim-to-real transfer increased from 75% to 90% as shown in Fig. 15. As shown in Fig. 16, we compare and discuss the generated trajectory with and without the error estimator. There is a clear gap between reality and simulation. The real target position is (0.25 m, 0.0 m, 0.06 m). Through error estimation, we can effectively correct the assembly trajectory in reality to improve the success rate of real assembly.

Our experiment snapshot is shown in Fig. 8. We can see that our control method can well transfer the policy from the simulation to reality. The robot moves more slowly as it gets closer to the target. This is

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

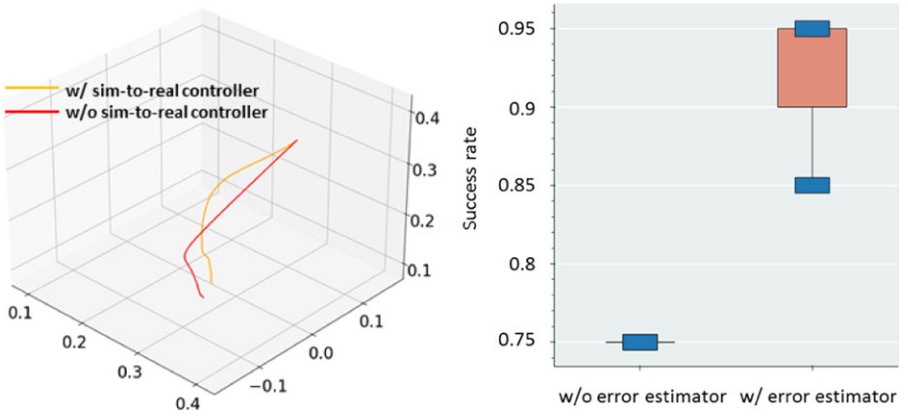

Figure 15. *The assembly trajectories and the average success rate of the sim-to-real transfer method.*

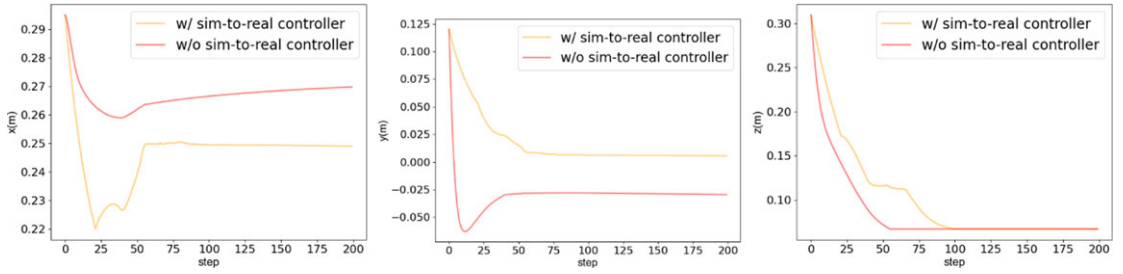

Figure 16. *Position changes of the sim-to-real transfer method.*

due to the setting of our reward function when the closer to the target is the smaller the variation of the reward, which helps to improve the safety of the assembly process.

6. Conclusion

In this paper, we proved that our framework is a novel approach to solving the assemble task both in simulated and real-world environments. We improved the PPO algorithm and compared it with the original method and other similar RL methods. The comparison results show that the proposed method can faster access to the assembly target with a higher success rate. With the visual demonstration, the PPO algorithm can generalize a more reliable and human-like policy and faster access to an optimal policy. Since there is a difference between reality and simulation, PPO modifies the domain parameters in the MuJoCo simulator while training. We also developed a general error monitoring method based on visual information to fill up the sim-to-real gap and increase the generalization of our framework. In this paper, PPO with visual demonstration and sim-to-real controller performs the gear assemble tasks well using UR3 robots with a higher success rate and fewer steps.

The approach contributes to overcoming the common limitations of data scarcity and high costs associated with real-world robot training. By combining the strengths of simulation-based learning, one-shot transfer policies, and the informative nature of visual demonstrations, the paper presents a pioneering solution to enhance the adaptability, efficiency, and performance of robotic assembly tasks. This research significantly advances the field by providing a comprehensive methodology that bridges the sim-to-real gap in robotic assembly through the incorporation of visual demonstrations and RL.

For future work, we will try to develop a more reliant and robust framework by joining the force information due to the significance of the force information in the assembly task. We will consider more assembly tasks and develop a more generalized framework for different assembly targets.

https://doi.org/10.1017/S0263574724000092 Published online by Cambridge University Press

Author contributions. Ruihong Xiao helped in conceiving the study concept and design, acquiring the data, analyzing, interpreting the data, and drafting the manuscript. Yiming Jiang and Hui Zhang helped in analyzing and interpreting the data and drafting the manuscript. Chenguang Yang helped in conceiving the study concept and design, drafting and revising the manuscript, obtaining funding, and supervising the study. All authors have read and agreed to the published version of the manuscript.

Financial support. This work was supported in part by National Nature Science Foundation of China (NSFC) under Grant U20A20200 and Major Research Grant No. 92148204, in part by Guangdong Basic and Applied Basic Research Foundation under Grants 2019B1515120076 and 2020B1515120054, in part by Industrial Key Technologies R&D Program of Foshan under Grant 2020001006308 and Grant 2020001006496.

Competing interests. The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential competing interests.

Ethical approval. None.

Supplementary material. The supplementary material for this article can be found at http://dx.doi.org/10.1017/S0263574724000092.

References

[1] H. Gao, Z. Li, X. Yu and J. Qiu, "Hierarchical multiobjective heuristic for pcb assembly optimization in a beam-head surface mounter," *IEEE Trans. Cybernet.* **52**(7), 6911–6924 (2021).
[2] S. Fujimoto, H. Hoof and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods" *International Conference on Machine Learning*, PMLR (2018) pp. 1587–1596.
[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous control with deep reinforcement learning" (2015), arXiv preprint arXiv: 1509.02971.
[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal policy optimization algorithms" (2017), arXiv preprint arXiv: 1707.06347.
[5] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel and S. Levine "Soft actor-critic algorithms and applications" (2018), arXiv preprint arXiv: 1812.05905.
[6] W.-C. Kwan, H.-R. Wang, H.-M. Wang and K.-F. Wong, "A survey on recent advances and challenges in reinforcement learning methods for task-oriented dialogue policy learning," *Mach. Intell. Res.* **20**(3), 318–334 (2023).
[7] R. Li and H. Qiao, "A survey of methods and strategies for high-precision robotic grasping and assembly tasks–some new trends," *IEEE/ASME Trans. Mechatron.* **24**(6), 2718–2732 (2019).
[8] N. Khlif, K. Nahla and B. Safya, "Reinforcement learning with modified exploration strategy for mobile robot path planning," *Robotica* **41**, 1–15 (2023).
[9] J. Dong, W. Si and C. Yang, "A novel human-robot skill transfer method for contact-rich manipulation task," *Robot. Intell. Automat.* **43**, 327–337 (2023).
[10] H. Qiao, Y.-X. Wu, S.-L. Zhong, P.-J. Yin and J.-H. Chen, "Brain-inspired intelligent robotics: Theoretical analysis and systematic application," *Mach. Intell. Res.* **20**(1), 1–18 (2023).
[11] H. Qiao, S. Zhong, Z. Chen and H. Wang, "Improving performance of robots using human-inspired approaches: A survey," *Sci. China Inf. Sci.* **65**(12), 1–31 (2022).
[12] S. Schaal, A. Ijspeert and A. Billard, "Computational approaches to motor learning by imitation," *Philos. Trans. R. Soc. Lond.. Ser. B Biol. Sci.* **358**(1431), 537–547 (2003).
[13] B. Wen, W. Lian, K. Bekris and S. Schaal, "You only demonstrate once: Category-level manipulation from single visual demonstration" (2022), arXiv preprint arXiv: 2201.12716.
[14] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations" (2017), arXiv preprint arXiv: 1709.10087.
[15] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards" (2017), arXiv preprint arXiv: 1707.08817.
[16] L. Zhang, Y. Feng, R. Wang, Y. Xu, N. Xu, Z. Liu and H. Du, "Efficient experience replay architecture for offline reinforcement learning," *Robot. Intell. Automat.* **43**(1), 35–43 (2023).
[17] W. Zhao, J. P. Queralta and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey" *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE (2020) pp. 737–744.
[18] K. Kimble, K. Van Wyk, J. Falco, E. Messina, Y. Sun, M. Shibata, W. Uemura and Y. Yokokohji, "Benchmarking protocols for evaluating small parts robotic assembly systems," *IEEE Robot. Automat. Lett.* **5**(2), 883–889 (2020).
[19] W. Wu, K. Liu and T. Wang, "Robot assembly theory and simulation of circular-rectangular compound peg-in-hole," *Robotica* **40**(9), 3306–3339 (2022).
[20] B. R. Lee and P. I. Ro, "Path finding and grasp planning for robotic assembly," *Robotica* **12**(4), 353–360 (1994).
[21] P. E. Hart, N. J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.* **4**(2), 100–107 (1968).

[22] A. Stentz, "Optimal and Efficient Path Planning for Partially-known Environments" *Proceedings of the 1994 IEEE international conference on robotics and automation*, IEEE (1994) pp. 3310–3317,
[23] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning" *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, IEEE, vol. **2** (2000) pp. 995–1001.
[24] L. Dong, Z. He, C. Song and C. Sun, "A review of mobile robot motion planning methods: From classical motion planning workflows to reinforcement learning-based architectures," *J. Syst. Eng. Electron.* **34**(2), 439–459 (2023).
[25] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya and R. Tachibana, "Deep Reinforcement Learning for High Precision Assembly Tasks" *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE (2017) pp. 819–825.
[26] E. Theodorou, J. Buchli and S. Schaal, "Reinforcement Learning of Motor Skills in High Dimensions: A Path Integral Approach" *2010 IEEE International Conference on Robotics and Automation*, IEEE (2010) pp. 2397–2403.
[27] S. Schaal, "Dynamic Movement Primitives-a Framework for Motor Control in Humans and Humanoid Robotics," **In:** *Adaptive Motion of Animals and Machines* (Springer, 2006) pp. 261–280.
[28] Z. Lu and N. Wang, "Dynamic movement primitives based cloud robotic skill learning for point and non-point obstacle avoidance," *Assembly Autom.* **41**(3), 302–311 (2021).
[29] L.-H. Kong, W. He, W.-S. Chen, H. Zhang and Y.-N. Wang, "Dynamic movement primitives based robot skills learning," *Mach. Intell. Res.* **20**(3), 396–407 (2023).
[30] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. P.Agapiou, J. Z. Leibo and A. Gruslys, "Learning from demonstrations for real world reinforcement learning" (2017).
[31] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.* **22**(1), 12348–12355 (2021).
[32] Y. Xu, C. Yang, X. Liu and Z. Li, "A Novel Robot Teaching System Based on Mixed Reality" *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*, IEEE (2018) pp. 250–255,
[33] Y. Xu, C. Yang, J. Zhong, N. Wang and L. Zhao, "Robot teaching by teleoperation based on visual interaction and extreme learning machine," *Neurocomputing* **275**, 2093–2103 (2018).
[34] Y. He, W. Sun, H. Huang, J. Liu, H. Fan and J. Sun, "Pvn3d: A Deep Point-wise 3D Keypoints Voting Network for 6dof Pose Estimation" *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020) pp. 11632–11641.
[35] S. Lin, Z. Wang, Y. Ling, Y. Tao and C. Yang, "E2EK: End-to-end regression network based on keypoint for 6d pose estimation," *IEEE Robot. Automat. Lett.* **7**(3), 6526–6533 (2022).
[36] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN" *Proceedings of the IEEE International Conference on Computer Vision* (2017) pp. 2961–2969.
[37] X. Wang, R. Zhang, T. Kong, L. Li and C. Shen, "Solov2: Dynamic and Fast Instance Segmentation," **In:** *Advances in Neural Information Processing Systems*. vol. **33**, (2020) pp. 17721–17732.
[38] S. Zakharov, I. Shugurov and S. Ilic, "Dpod: 6D Pose Object Detector and Refiner" *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019) pp. 1941–1950.
[39] R. B. Rusu, N. Blodow, Z. C. Marton and M. Beetz, "Aligning Point Cloud Views Using Persistent Feature Histograms" *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE (2008) pp. 3384–3391.
[40] R. B. Rusu, N. Blodow and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D Registration" *2009 IEEE International Conference on Robotics and Automation*, IEEE (2009) pp. 3212–3217.
[41] P. J. Besl and N. D. McKay, "Method for Registration of 3-D Shapes," **In:** *Sensor Fusion IV: Control Paradigms and Data Structures*. vol. **1611** (Spie, 1992) pp. 586–606.
[42] Y. Chen, C. Zeng, Z. Wang, P. Lu and C. Yang, "Zero-Shot sim-to-real transfer of reinforcement learning framework for robotics manipulation with demonstration and force feedback," *Robotica* **41**, 1015–1024 (2022).
[43] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba and P. Abbeel, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World" *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE (2017) pp. 23–30,
[44] K. Arndt, M. Hazara, A. Ghadirzadeh and V. Kyrki, "Meta Reinforcement Learning for Sim-to-Real Domain Adaptation" *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE (2020) pp. 2725–2731.
[45] Y. Li, J. Song and S. Ermon, "Infogail: Interpretable Imitation Learning From Visual Demonstrations," **In:** *Advances in Neural Information Processing Systems*. vol. **30** (2017).
[46] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning (2016).
[47] E. Todorov, T. Erez and Y. Tassa, "Mujoco: A Physics Engine for Model-based Control" *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE (2012) pp. 5026–5033.
[48] J. Collins, S. Chand, A. Vanderkop and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access* **9**, 51416–51431 (2021).
[49] J. Zhang, M. Li, Y. Feng and C. Yang, "Robotic grasp detection based on image processing and random forest," *Multimed. Tools Appl.* **79**(3), 2427–2446 (2020).

Cite this article: R. Xiao, C. Yang, Y. Jiang and H. Zhang (2024). "One-shot sim-to-real transfer policy for robotic assembly via reinforcement learning with visual demonstration", Robotica **42**, 1074–1093. https://doi.org/10.1017/S0263574724000092