Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

Robot Control in Human Environment using Deep Reinforcement Learning and Convolutional Neural Network

Chen Chen[1*], Hsieh-Yu Li[2*], Audelia G. Dharmawan[2], Khairuldanial Ismail[2], Xiang Liu[1], and U-Xuan Tan[2], *Member, IEEE*

***Abstract*— Deep reinforcement learning (DRL) has been employed in numerous applications where complex decision-making is needed. Robot control in a human environment is an example. Such algorithm offers possibilities to achieve end-to-end training which learns from image directly. However, training on a physical robotic system under human environments using DRL is inefficient and even dangerous. Several recent works have used simulators for training models before implementing to physical robots. Although simulation provides efficiency to obtain DRL trained models, it poses challenges for the transformation from simulation to reality. Since a human environment is often cluttered, dynamic and complex, the policy trained with simulation images is not applicable for reality. Therefore, in this paper, we propose a DRL method to achieve end-to-end training in simulation, as well as to adapt to reality without any further finetune. Firstly, a Deep Deterministic Policy Gradient algorithm (DDPG) is employed to learn policy for robot control. Secondly, a pre-trained Convolutional Neural Network algorithm (CNN) is used to visually track the target in image. This technique provides the efficient and safe DRL training in simulation while offering robust application when a real robot is placed in dynamic human environment. Simulation and experiment are conducted for validation and can be seen in the attached video. The results have shown successful demonstration under various complex environments.**

I. Introduction

Coupled with deep neural networks, reinforcement learning (RL) algorithm has more powerful ability to learn satisfactory policies. The deep reinforcement learning (DRL) is beneficial to the situations of complex environment and that of large number of decision-making. For instance, in Atari games [1], Deep Q Network take pixel as input. Or in game of Go [2], DRL has achieved superhuman performance.

There are other fields where complex environment and large number of decision-making are also needed, such as robot control. Two main challenges arise when DRL is applied for robot control. The first challenge is that training on robots in reality is slow and inefficient as illustrated in Fig. 1 (A). This is because a large number of samples is often needed to be explored in a high-dimensional continuous state space [3]. To make matter worse, if DRL is trained for a robot that is used in a human environment, the safety is not well guaranteed since the algorithm might diverge.

[1]C. Chen, and X. Liu are with School of Software and Microelectronics, Peking University, China (email: chrisole@pku.edu.cn, xliu@ss.pku.edu.cn).
[2]H.-Y. Li, A.G. Dharmawan, K. Ismail, and U-X. Tan are with Pillar of Engineering Product Development, Singapore University of Technology of Design, Singapore (email: hsiehyu_li@sutd.edu.sg, audelia@sutd.edu.sg, Khairuldanial@mymail.sutd.edu.sg, uxuan_tan@sutd.edu.sg).
[*]Equal contribution

A recent approach to circumvent such challenges is to use a simulation platform. Samples in simulation environment are typically easier to be obtained than real environment [3], without the limit relating to acquisition. Moreover, faster training in simulation and the feature of training parallelization lead to higher effectiveness as compared to obtaining data in real environment. Most importantly, there are no accidental damages in the simulation environment where human safety is ensured.[4]

However, this results in another challenge which is the transfer from simulation to the real world as shown in Fig. 1 (B). After finishing end-to-end training with simulation images, the policy is not able to work directly on a real robot. This is because the reality environment of robot is usually messy and complex, which leads to difference between images of simulation and reality. When reality images are taken as input, the policy is not able to complete the task trained in simulation [4].

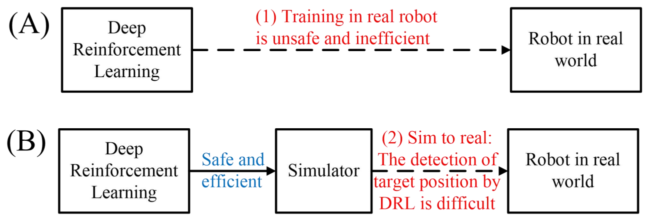
(A)
Deep Reinforcement Learning
(1) Training in real robot is unsafe and inefficient
Robot in real world
(B)
Deep Reinforcement Learning
Safe and efficient
Simulator
(2) Sim to real: The detection of target position by DRL is difficult
Robot in real world

Fig. 1 Research problem. (A) It is unsafe and inefficient to directly train the model in a real robot. (B) Training on a simulator using DRL is promising but the challenges pose on the transfer from simulation to reality, especially when the robot is placed in a human environment.

Therefore, we propose a method to achieve end-to-end training in simulation, and simultaneously to adapt to reality. A simulation environment is built where the DRL is trained to reach the target position. Additionally, we employ a CNN algorithm of object detection [5] to pay attention on this part, then offer position information for agent. Since only the target is the interested part of the whole image (both in simulation and reality), this provides the robustness in transferring from simulation to a real robot. This method also provides the flexibility of the tasks. If the tasks or targets are changed, it can be simply modified and retrained in the simulator. Moreover, the tasks can be robustly executed in a physical robot under various complex environments. This contribution will make DRL more applicable in the field of robotics.

II. Background and Related Work

A. Deep Reinforcement learning

Reinforcement learning has been applied in the tasks of robot in control, locomotion [5] or manipulation [6], [7]. To solve the issue of limited decision space for traditional RL, the function approximators, such as neural networks, have been used with RL [8]. With deep leaning, versatile tasks in robotics, which are difficult to solve using traditional control approach [9], can be accomplished. However, when deep neural network is used as function approximators, it becomes potentially unstable when RL combined with a value-based type recursion [10].

To address the issue of potentially unstable trained results, a Deep Deterministic Policy Gradient (DDPG) algorithm is proposed in [11]. DDPG is one of the DRL algorithms that are designed for continuous control tasks. DDPG is based on Actor-Critic network and it combines the advantages of Deep Q-Network (DQN), which can make continuous decision in action space making it suitable for manipulator control. Studies [9], [12] have shown that the structure of DDPG can be adjusted flexibly to accomplish different tasks.

B. Sim-to-real

The transfer from simulation to real world, or sim-to-real, is an important step when DRL algorithm is applied to reality. This is considered an instance of domain adaptation, migrating the trained source domain (sim) to the new target domain (real). Progressive networks have also been used to transfer policies of a robotic arm from simulation to the real world [13]. However, training is still needed in the physical robot arm. There are also dynamic randomization methods [14], in which numerous trainings are carried out in the simulation so that the algorithm can be directly adapted to the real physical system.

To interact with real world, image are useful sensing data that can be acquired in cameras. Existing research in DRL aim to learn from pixel to achieve end-to-end training. A DQN learning from images is employed in [15], the robot reaches the target position which is obtained by synthetic images. However, it failed to transfer the network to real hardware by raw-pixel images because the details of the real world are under-modeled if synthetic scenes are used [16]. Randomization in the visual domain is a promising method to be used to directly transfer vision-based policies from simulation to the real world [17], [18]. This method eliminates the need of real images during training. Researchers in [12] randomize some visual element for simulation images as actor network input but a large amount of randomization and training resources are needed. The drawback of these methods for DRL is the results are affected by the complexity of the environment. Therefore, the proposed method in this paper aims to address this issue, which is discussed in the next section.

III. Proposed Method

The proposed method is as shown in Fig. 2. DDPG is employed to train the model in simulation to accomplish tasks. The policy is subsequently transferred in a physical robot with CNN algorithm, which is aimed for target detection.

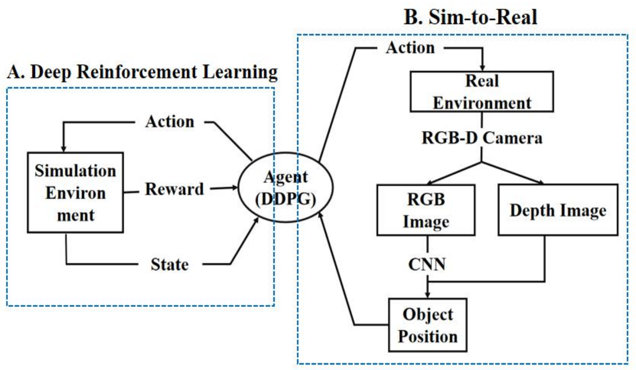

Fig. 2 The proposed method. DRL is used for robot motion control (part A). Mask R-CNN is used for target detection. Combing the trained model with Mask R-CNN (part B), the task is executed in real robot.

A. Deep Reinforcement learning

DRL is a type of machine learning where an agent interacts with environments to maximize the sum of rewards. The rewards are defined according to the requirement of different tasks. In this paper, we design a reaching task for a robot arm as an illustration. The details of DRL for this reaching task are as follows.

1. State, Action and Reward

The current state (S_t) is the information that agent senses in current environment. In robot reaching task, joint angles and velocities are usually considered as S_t. In this paper, besides position of the corresponding robot joints, the position of goal is also considered in S_t.

To achieve the next state (S_{t+1}), an action (A_t) needs to be executed. Six robot joint velocities are defined as A_t. The robot interacts with the environment based on the action.

The reward, R, defines what are the good and bad events for the agent [19] and therefore the action is also evaluated. There are two main methods to define rewards for robot control. The first one is to a sparse reward [12]. The benefit of the sparse reward is that a better policy (π) is often yielded with appropriate training [12], [20]. Though it often creates promising policies, discovering a sparse reward is a hard exploration [21] as it normally takes time to obtain the results. The second method is using a smoothly varying reward [9]. On the contrast to the first method, a more efficient training process is usually obtained in the second method. Therefore, we use a smoothly varying reward in our algorithm. To define a smoothly varying reward, the distance between end-effector and object is firstly represented as D. The opposite of the distant, $-D$, is then defined as the reward. In other words, when the distance of end-effector and the target is shorter, the reward increases.

2. Details of DDPG

DDPG is a model-free algorithm for continuous action space. Compared to deterministic policy gradient algorithms (DPG), DDPG is improved in two aspects: target network and experience reply [11]. The target networks for actor and critic are computed on separate versions of actor and critic networks [12], which change at a slower rate than the main networks. On the other hand, experience reply is learned from

Deep Q-learning Network algorithm. A replay buffer is built. Transitions $\{S_t, A_t, R, S_{t+1}\}$ are collected into the buffer [9] for each time step. The algorithm is shown as part A in Fig. 2 and as follows.

```
Algorithm DDPG [11]
  Initialize replay buffer $R_b$
  Initialize critic network $Q(S, A|\theta^Q)$ and actor $\mu(S|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$
  Initialize target network $Q'$ and $\mu'$
  for episode = 1, $M$ do
    Initialize a random goal and render as $I_0$
    get state $S_0$ from $I_0$
    for $t$=1, $T$ do
      Select action according to the current policy and noise $N_t$:
      $A_t \leftarrow \pi(S_t) + N_t$
      Execute action $A_t$ to environment
      Receive reward $R$ and transition to $S_{t+1}$
      Store $\{S_t, A_t, R, S_{t+1}\}$ in $R_b$
    end for
    for $i$=1, $I$ do
      Sample a random minibatch of $n$ $\{S_t, A_t, R, S_{t+1}\}$ from $R_b$
      Set discount factor $\gamma$
      Set $Y_i = R_i + \gamma Q'(S_{i+1}, \mu'(S_{i+1}|\theta^{\mu'})|\theta^{Q'})$
      Update critic network by minimizing the loss:
      $L = 1/n \sum_i (Y_i - Q(S_i, A_i|\theta^Q))^2$
      Update actor policy using the sample policy gradient:
      $\nabla\theta^\mu J \approx 1/n \sum_i \nabla_A Q(S, A|\theta^Q)|_{S=S_i, a=\mu(S_i)} \nabla\theta^\mu \mu(S|\theta^\mu)|_{S_i}$
      Update the target networks
    end for
  end for
```

When DDPG is applied for reaching task in simulation, there are few details that need to be illustrated. Firstly, A_t is computed from the actor network with random initialization θ^μ which are applied to the environment to get S_{t+1} and R as feedbacks. Secondly, the transition tuples $\{S_t, A_t, R, S_{t+1}\}$ are stored in the replay buffer. When the replay buffer is full, minibatch size groups of data are randomly extracted for training. The transition tuples $\{S_t, A_t, R, S_{t+1}\}$ are still stored to update replay buffer one by one. This approach offers abundant data that can be unlimited obtained. Thirdly, with the training going on, the actor commands generated by actor network ensure that the requirements of the tasks are fulfilled by maximizing reward. As a result, the end-effector gets closer to the target. Fourthly, after training, the model needs to be tested. Given a random S_t, if the robot is able reach the target using A_t that is obtained by actor network, it means that the task is executed successfully using the trained model. A successful task is defined by the users and in this paper, it is defined as a minimum distance. If the end-effector reaches within this minimum distance, the training is considered complete. Finally, the recorded joint angles from the test in simulation are applied to the real robot arm.

B. *Sim-to-real*

Images taken from human environments are complex and unstructured. To make matter worse, human behavior is unpredicted that causes the changes of environment.

As shown in part B in Fig. 2, an open source algorithm – Mask R-CNN [22] – is used in this paper. The features of Mask R-CNN are fast detection speed and high recognition accuracy. It can also be easily adjusted by adding training sets when new detection targets are needed. Most importantly, given complex and messy background, the target is still robustly detected using this algorithm [22]. Based on these features, target detection in human environments benefits from Mask R-CNN and therefore, it is employed in this method both in simulation and reality.

A finetuned Mask-RCNN model is added in training part, while the weight is fixed. Coupled with depth images, the 3D information of target together with state of robot joint are fed into actor network. One thing is noted that the position of target is fixed during one training episode, it is unnecessary and time-consuming to run object detection in every timestep. We do that at the beginning of one episode. After adequate training, the robot learns reasonable policy from images and its joint. In the real world, a RGB-D camera with hand-eye calibration [23] [24] is employed to offer same pixel size images contain target. The position of the target with respect to the world coordinate is calculated by the pin-hole camera model. Then trained actor network is used to find the robot joint commands from the previous sub-section. These robot joint commands are then inputted in a real robot to generate the robot movement.

IV. SIMULATION AND EXPERIMENT

To evaluate the effectiveness of transfer sim-to-real using the proposed method, simulation and experiments of robot reaching task are conducted in this section. The simulation consists of 2 parts: training and testing. The experiments also consist of 2 parts: robot in different environments and target with other disturbances.

A. *Simulation*

A robot arm model of UR10 is in the Mujoco simulation environment. This robot arm consists of six degree-of-freedom (DOF). These 6 DOF joint angles must be obtained in every time steps and can be controlled to drive the robot to the desired pose. A ball is set as the target for UR10 to reach. This target is placed in a random location within the workspace of UR10. The location is fixed within one training episode, but it is randomly changed from episode to episode.

1. *Training in simulation*

There are 500 timesteps in one training episode. Object detection is called before first timestep and generate the position of target. An action is executed during each time step and a group of $\{S, A, R, S'\}$ is recorded. When an episode ends after 500 timesteps, the robot arm goes to home position with a new location of the target. A new episode starts after that. Moreover, the speed of each joint is usually limited within a boundary in simulation because an unconstrained speed might cause the unstable motion.

The reward and the minimal distance of each episode are recorded to observe the learning process. For better visualization, we plot the results every 30th episode as shown in Fig. 3 and Fig. 4. As shown in Fig. 3, the first 200 episode is used to store buffer, and the change of reward is flat. At this moment, the learning is not started where the robot arm randomly moves. After 200 episodes, the reward (Fig. 3) begins to increase and the minimal distance between the target and robot end-effector gradually decreases (Fig. 4). After 1000 episode, the change of the minimal distance becomes stable which demonstrates the learning process is complete.

There are a variety of factors that affect learning effectiveness. Firstly, for neural network, the effectiveness of convergence for the training significantly depends on learning rate and the size of networks. Additionally, the initial value of A_t might also affect the learning effectiveness. This is because a large initial value of the action drives the robot away from the target. It becomes harder to achieve the target position given limited time steps. A resolution for such a situation is to limit the initial action range with a small value. Lastly, the reward function also affects both learning efficiency and learning result. To accelerate the learning process, extra rewards can be given when the distance is smaller than a certain value. For example, if the distance decrease to 0.1m, the reward is increased by an extra value (0.5 in our experiment).

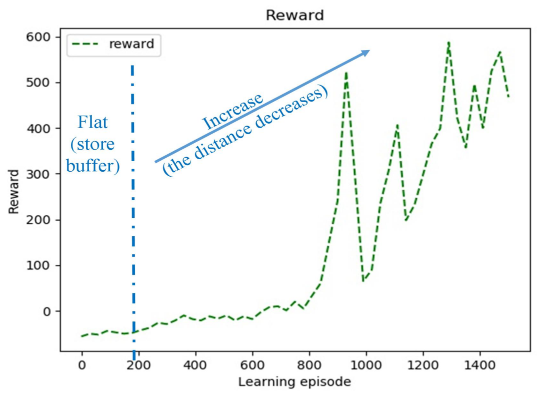

Fig. 3 The change of the reward during training (it is plotted every 30th episode).

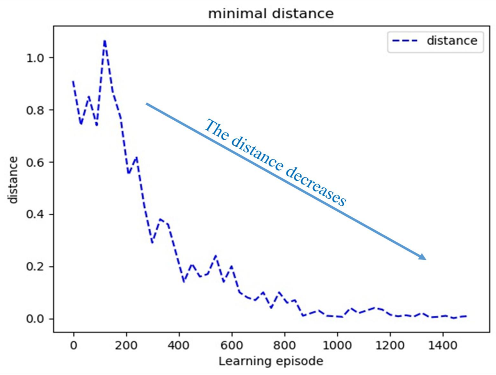

Fig. 4 The change of the minimal distance during training (it is plotted every 30th episode). The tendency of the change is opposite to the change of the reward. In other words, when the rewards increase (Fig. 3), the distance decreases (Fig. 4), which is the requirement of the reaching task.

1. *Testing in simulation*

After the training, the trained model needs to be tested in simulation. The agent (UR10) is tested for 10 episodes and each test also take 500 timesteps. The results of minimal distance for 10 episodes are shown in Fig. 5. With the trained model, the robot is able to move to any random target position. Most of the errors are less than 0.01m as shown in Fig. 5.

The result of one of the episodes is shown in Fig. 6. It shows the change of the distance between the target and robot end-effector for 500 timesteps. The robot end-effector moves to the target at the distance of less than 0.015m within 250 episodes. This demonstrates the robot can successfully reach the target using the trained model. Note that even though the robot movement fluctuates after it reaches the distance of less than 0.015m, it can be solved by down sampling the commands to the real robot.

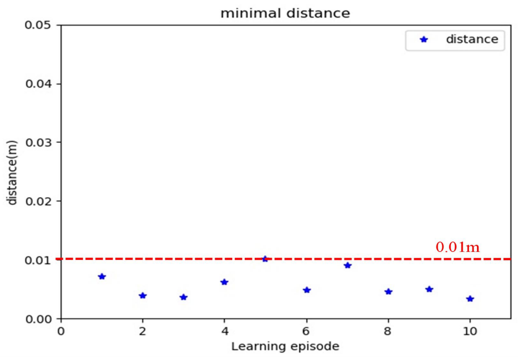

Fig. 5 The minimal distance in 10 episode

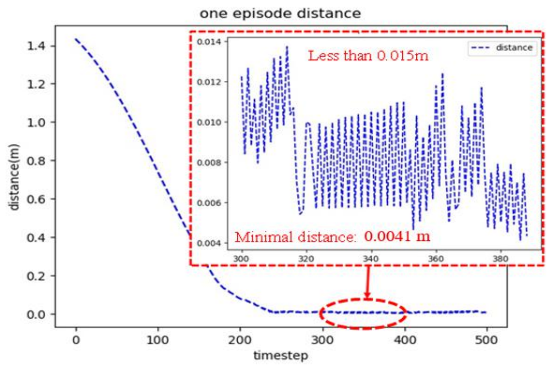

Fig. 6 The distance in one episode of testing.

B. *Experiments in a real robot*

The experimental setup is as shown in Fig. 7. UR10 (Universal Robotic Inc., Denmark) is used to accomplish a task of reaching a yellow ball. A calibrated Kinect 2 camera (Microsoft Corporation, USA) is employed for RGB-D images. Kinect 2 is fixed in front of the robot base. This robotic system is placed in a laboratory where the environment is disordered and messy. Moreover, since the target detection is robust in our method, the target (yellow ball) can be simply put on a chair without the need to be hung in front of a clean curtain like traditional setup for DRL [12], [15]. Additionally, color of the target (yellow) is also randomly chosen without the need of choosing same color in the simulation (red). The experimental results are also demonstrated in the attached video.

The procedure of sim-to-real is as follows. In the real image with same pixel size of simulation image, the target is detected by Mask R-CNN and is transferred into the position with respect to world coordinate using hand-eye calibration. With the trained model by DRL in the previous section, the target position is tested in the simulation. A series of actions through the simulation is generated to drive the end-effector to reach the target position. If the distance between the target and end-effector is less than 0.015m, the episode ends. The robot joint commands are subsequently obtained from the simulation at equal time step. For instance, if the end-effector reaches the minimum distance within 200 timesteps, 20 robot joint commands can be obtained at every 10 time steps. Finally, the obtained robot command is implemented into real UR10. To reach the target, the end-effector is supposed to move to the center of the ball. However, to avoid the obstacle (such as chair) collision in implementation, we change the location of the target of 2 cm directly above the top of the ball.

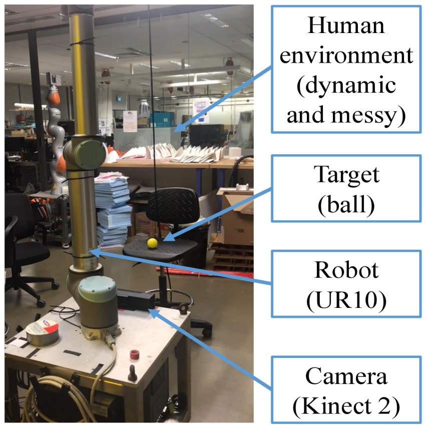

Fig. 7 The experimental setup.

1. Robot in different environments

To verify the robustness of target detection in dynamic human environment, the first experiment is conducted as shown in Fig. 8.

The experiment includes three set of comparison, namely, clean background, messy background and another messy background with different robot location. A clean curtain is used in the first set of experiment to provide a relative simple background for input image. The second set experiment is conducted without the curtain where the background becomes messy. Another messy background is setup in the third set of experiment by changing the robot location. Additionally, in the left side of Fig. 8, the image that is obtained from Kinect 2 is shown. Mask R-CNN results for the target yellow ball are shown in green bounding box. The robot movement to reach the target is illustrated at the right side of Fig. 8. With the proposed method, the result from simulation is successfully transferred to reality. The robot reaches the location of the target under human environments which are messy and dynamic.

2. Target with other disturbances

In this experiment, the robustness of target detection where there exists other disturbance is evaluated as shown in Fig. 9. The experiment also contains three set of comparison, namely, the target is placed in front of same color (yellow) background, there is another disturbance with the same shape (red ball) and the target is partially blocked by a book. All the targets are detected in Fig. 9 are detected and the robot successfully reaches to the corresponding position using the commands from DRL.

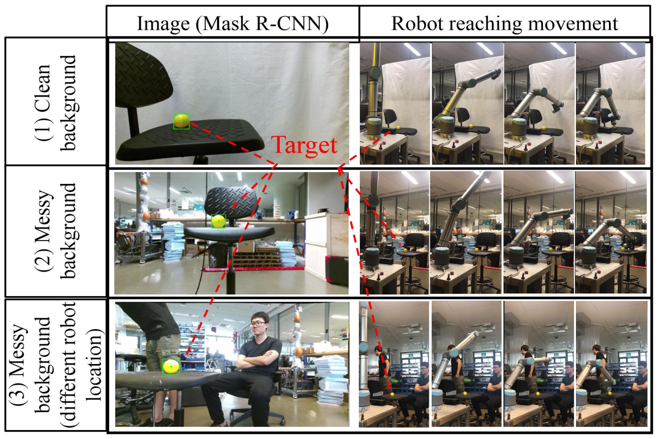

Fig. 8 The results of first experiment (3 sets). Left: the image from camera (the green box is detected by Mask R-CNN). Right: robot movement. (1) When the background is clean, (2) when the robot is placed in a human environment and, (3) in another messy environment with a different robot location (human is inside the image).

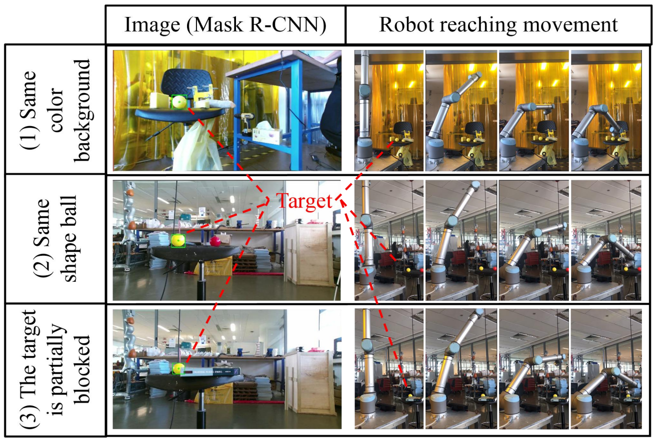

Fig. 9 The results of second experiment (3 set). Left: the image from camera (the green box is detected by Mask R-CNN). Right: robot movement. (1) Background is the same color (yellow) with the target ball, (2) there is another red ball with same shape of the target and, (3) the target is blocked by a book.

C. *Discussion*

We trained a model with DDPG algorithm to finish reaching task in simulation. After 3 hour training, error of testing is usually less than 0.015m. This shows one of the advantages in testing in simulation, namely, the error is accurately and easily obtained, which can be troublesome in a real robot. Another advantage of test in simulation is that the motion process of robot arm can be observed in simulation before commands are executed in real physical robots. Such advantage provides the safety as an incorrect behavior can be seen in the simulation and hence it can be avoided in reality.

Fig. 8. shows that the positions of yellow ball are detected using Mask R-CNN in three experiments subject to clean, messy or dynamic background. The confidence of the corresponding bounding box for the target is still high ((1) 1.00, (2) 0.999 and (3) 0.928) in the messy background. In other word, the target can be accurately detected. One thing to take note is that even though the confidence in the third set is relatively lower (0.928) due to the disturbance from the human behind, it is still robustly recognized.

For Fig. 9, in first set of experiment, the target can be detected accurately in present of same color background. In the second set where the disturbance is the same shape as the target, the procedure using Mask R-CNN can be simply modified with an extra color detection. This provides the flexibility when defining the target in the image. Lastly, if the target is partially blocked (3rd set), yellow ball still can be located with a lower confidence ((1)0.976, (2)0.974 and (3) 0.938) which shows the robustness of the algorithm.

The important advantage of this method using DRL and CNN is that it provides great extensibility to design more complex tasks such as dexterous manipulation. It is promising because firstly, design of rewards of DRL are flexible. Secondly, efficient training for the next task can be achieved by restoring the parameters the current tasks. Lastly, various objects can be detected simultaneously using Mask R-CNN, which is beneficial for tasks with multiple targets.

V. Conclusion

In this paper, a method using Deep Deterministic Policy Gradient algorithm and Mask R-CNN for robot control in human environments is proposed. A simulation is setup on Mujoco platform where an end-to-end policy is trained using DDPG algorithm from images. The trained policy is implemented directly in a physical robot UR10 to make decision without any further training. A target reaching task is carried out in experiment where the environments are dynamics and messy. The experimental results show that the robot accurately locates and reaches the target position by the proposed method.

By proposing the separation of DRL in simulation and target detection in reality, we hope to contribute towards the flexibility of the tasks, the robustness of target detection under dynamic environments and realization of robot complex behavior. In this paper, the task where the robot reaches the target is investigated while this method can be further extended to more complex requirements such as manipulation or grasping. The long-term goal is to open up new possibilities in the ways the robotic system can be safely used in human environments for versatile tasks.

References

[1] V. Mnih and D. Silver, "Playing Atari with Deep Reinforcement Learning," *arXiv:1312.5602v1 [cs.LG]*, 2013.

[2] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7585, pp. 484–489, 2016.

[3] M. Cutler and J. P. How, "Efficient Reinforcement Learning for Robots using Informative Simulated Priors," *2015 IEEE Int. Conf. Robot. Autom.*, pp. 2605–2612, 2015.

[4] C. Gaz, E. Magrini, and A. De Luca, "A model-based residual approach for human-robot collaboration during manual polishing operations," *Mechatronics*, vol. 55, no. June 2017, pp. 234–247, 2018.

[5] N. Kohl and P. Stone, "Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion," in *The International Conference on Robotics and Automation*, 2004, no. April.

[6] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement Learning of Motor Skills in High Dimensions : A Path Integral Approach," in *International Conference on Robotics and Automation*, 2010, vol. 1, no. 3, pp. 2397–2403.

[7] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning Force Control Policies for Compliant Manipulation," *2011 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 4639–4644, 2011.

[8] R. Hafner, M. Riedmiller, and A. Overall, "Neural Reinforcement Learning Controllers for a Real Robot Application," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007.

[9] I. Popov *et al.*, "Data-efficient Deep Reinforcement Learning for Dexterous Manipulation," *arXiv Prepr. arXiv1704.03073*, no. section V, 2017.

[10] V. François-lavet, R. Fonteneau, and D. Ernst, "How to Discount Deep Reinforcement Learning : Towards New Dynamic Strategies," in *European Workshop on Reinforcement Learning (EWRL)*, 2016, pp. 1–9.

[11] J. J. Hunt *et al.*, "Continuous learning control with deep reinforcement," in *International Conference on Learning Representations (ICLR)*, 2016.

[12] R. O. Oct, "Asymmetric Actor Critic for Image-Based Robot Learning," *arXiv:1710.06542v1 [cs.RO]*, 2017.

[13] A. A. Rusu, M. Veˇ, T. Rothörl, and N. Heess, "Sim-to-Real Robot Learning from Pixels with Progressive Nets," *arXiv Prepr. arXiv1610.04286*, 2016.

[14] R. O. Mar, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," *arXiv:1710.06537*, 2018.

[15] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control," in *Australasian Conference on Robotics and Automation*, 2015.

[16] R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-fei, and A. Farhadi, "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning ∗," in *The IEEE International Conference on Robotics and Automation*, 2017.

[17] F. Sadeghi, "CAD 2 RL : Real Single-Image Flight Without a Single Real Image," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018, pp. 4691–4699.

[18] 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," in *In the proceedings of the IEEE/RSJ International Conference on Intelligent RObots and Systems (IROS)*, 2018.

[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning : An Introduction*. 2017.

[20] M. Andrychowicz *et al.*, "Hindsight Experience Replay," in *. In the Annual Conference on Neural Information Processing Systems (NIPS)*, 2017.

[21] M. Riedmiller *et al.*, "Learning by Playing – Solving Sparse Reward Tasks from Scratch," *arXiv Prepr. arXiv1802.10567*, 2018.

[22] P. Doll, R. Girshick, and F. Ai, "Mask R-CNN," in *CoRR, abs/1703.06870*, 2017.

[23] R. Horaud and F. Dornaika, "Hand-eye Calibration," *Int. J. Rob. Res.*, vol. 14, no. 3, pp. 195–210, 2011.

[24] T. Wiedemeyer, "IAI Kinect2," *Institute for Artificial Intelligence, University Bremen*, 2015. [Online]. Available: https://github.com/code-iai/iai_kinect2.