Mixed-Reality Deep Reinforcement Learning for a Reach-to-grasp Task [⋆]

Hadi Beik Mohammadi[1] [0000−0002−8170−2471] ∗, Mohammad Ali Zamani[1], Matthias Kerzel[1], and Stefan Wermter[1]

Knowledge Technology, Department of Informatics, University of Hamburg, Germany
6beik / zamani / kerzel / wermter @informatik.uni-hamburg.de
http://www.knowledge-technology.info

Abstract. Deep Reinforcement Learning (DRL) has become successful across various robotic applications. However, DRL methods are not sample-efficient and require long learning times. We present an approach for online continuous deep reinforcement learning for a reach-to-grasp task in a mixed-reality environment: A human places targets for the robot in a physical environment; DRL for reaching these targets is carried out in simulation before actual actions are carried out in the physical environment. We extend previous work on a modified Deep Deterministic Policy Gradient (DDPG) algorithm with an architecture for online learning and evaluate different strategies to accelerate learning while ensuring learning stability. Our approach provides a neural inverse kinematics solution that increases over time its performance regarding the execution time while focusing on those areas of the Cartesian space where targets are often placed by the human operator, thus enabling efficient learning. We evaluate reward shaping and augmented targets as strategies for accelerating deep reinforcement learning and analyze the learning stability.

Keywords: Deep reinforcement learning · visuomotor learning · neurorobotic models.

1 INTRODUCTION

Deep continuous reinforcement learning enables physical robots to acquire visuomotor abilities by interacting with their environment. Following the paradigm of developmental robotics, this would enable the robots to master more and more challenging tasks in complex environments [3]. However, deep reinforcement learning depends on extended periods of trial-and-error learning [10] that expose robots and their environment to physical stress. Virtual environments and simulations overcome this challenge by offering fast and safe training, but transferring learned abilities from simulation to the real world can be challenging. Furthermore, neural deep reinforcement learning approaches can struggle

[⋆] The authors gratefully acknowledge partial support from the German Research Foundation DFG under project CML (TRR 169) and the European Union under project SECURE (No 642667).

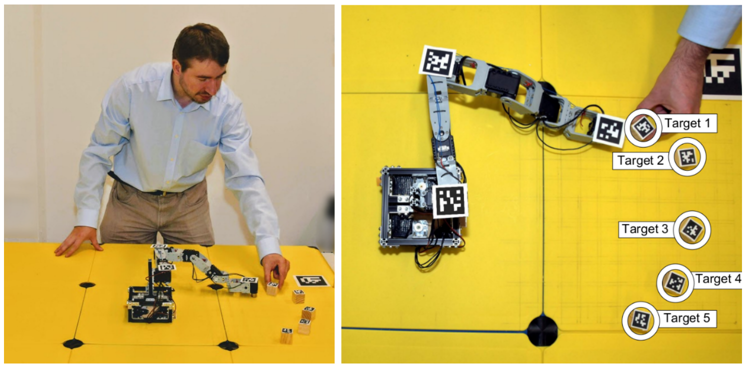

Fig. 1. Learning setup: (left) A robot arm with two degrees of freedom for planar movement uses continuous deep reinforcement learning to reach for objects marked with QR codes. Our architecture extracts the environmental information including the robot's joint length and target position, and learns to solve the specific reach-to-grasp task in its simulator. Once the learner is confident about the solution, it is deployed on the real robot. (right) Experimental setup: The robot repeatedly reaches for the targets T_1 to T_5, always in the same order. We evaluate the learning speed and the stability of different deep reinforcement learning strategies.

with stability [13] and catastrophic forgetting [12], i.e., "unlearning" of acquired skills through new experiences.
To address these issues we propose an architecture for online learning: usually, reach-to-grasp abilities are trained with a large number of random positions. In contrast, our approach focuses on online learning of interactively provided targets, thus avoiding extensive initial training times and attention to unused parts of the workspace. We evaluate different strategies for fast and stable learning based on a modified Deep Deterministic Policy Gradient (DDPG) [11] algorithm. We extend previous work on task simplification in a virtual simulation [9] with an integrated mixed-reality architecture that allows task-specific learning. In our experimental setup, an experimenter provides a visual target to the robot. The geometry of the arm and the target are transferred to a simulation model, where solving the specific task instance is learned. This solution is then executed with the real arm, as depicted in Figure 1 (left). The entire trial-and-error learning phase is carried out in simulation; only the final result is executed on the real robot. One advantage of this approach is that, over time, past experiences enhance the performance of the system leading to faster results as learning is progressing. We present an interactive mixed-reality experimental setup and a modified neural architecture for deep continuous reinforcement learning that takes into account task-focused learning. We evaluate different strategies to accelerate the simulated learning process. Experimental results demonstrate the ability of the approach to produce correct reach-to-grasp solutions and perform stable learning from past experiences over time.

2 BACKGROUND AND RELATED WORK

2.1 Inverse Kinematics

Inverse kinematics is defined as finding a joint configuration $\Theta = \Theta_{i=1,\ldots,n}$ that moves the end effector of a robotic arm into a specified pose, e.g., reaching for a graspable object. Though this problem can be solved analytically for a small number of degrees of freedom (DoFs), it becomes intractable when the complexity of the robotic arm increases. Therefore, state-of-the-art inverse kinematics approaches utilize pseudoinverse Jacobians to minimize the pose error of a current joint configuration. Strategies like random restarts are used to overcome local minima; this is for instance used by the inverse kinematics solver TRAC-IK [2]. In contrast, solvers based on feed-forward neural networks offer constant computation time. However, they require a large amount of training data and can suffer from relatively high pose errors [8, 5]. These issues can be addressed with deep reinforcement learning.

2.2 Continuous Deep Reinforcement Learning

Reinforcement learning is based on trial-and-error explorations. In the past, tabular Q-learning [18] approximates the optimal state-action value function for an agent which interacts with its environment. The policy of the agent can converge to an optimal policy which maps the given state s_t to the action a_t for which the expected return is the maximum. Tabular Q-learning was limited to a discrete state and action representation; it had no possibility for generalization over states and actions. Mnih et al. [13] introduced Deep Q-Network learning, which applies function approximation to use high dimensional raw input data, e.g., unprocessed pixel data as the state, thus enabling q-learning in a continuous state space. However, it is still limited to discrete actions.

Finally, the DDPG introduced by Lilicrap et al. [11] is an actor critic RL which uses two separate neural networks to predict the return of an action (critic: $Q(s_t, a_t; \theta^Q)$ and to suggest optimal actions (actor: $\mu(s_t; \theta^\mu)$). Like DQN, the DDPG agent uses a memory replay to record its experience, *state*, *action*, *next state* and *reward* (s_t, a_t, s_{t+1}, r_t). Since, direct using of the actor and critic networks may cause instability in target value, two target networks, actor $Q'(s_t, a_t; \theta^{Q'})$ and critic $\mu'(s_t; \theta^{\mu'})$ were used. The target networks are updated gradually toward the primary actor and critic networks. In each update, a batch of experience is sampled and the target networks are used to calculate the target value

$$y_t = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}; \theta^{\mu}{}'); \theta^{Q'}) \tag{1}$$

to train the critic network and its parameters by optimizing the following loss.

$$\min_{\theta^Q} \frac{1}{N} \sum_i (y_t - Q(s_t, a_t; \theta^Q))^2 \tag{2}$$

The actor network parameter is updated using the sampled policy gradient.

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i}^{a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i} \tag{3}$$

After updating both networks, the target network parameters are slowly updated

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'},\ and\ \theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \tag{4}$$

where $\tau \ll 1$ is the weight to slowly transfer the parameters from trainable models to a target model to gain more stability [11].

Though the proposed mixed-reality architecture can be applied to most variants of (continuous) deep reinforcement learning algorithms like the continuous actor critic learning algorithm (CACLA) [17] and continuous deep q-learning with model-based acceleration [6], we focus on the widely-used and well-established DDPG algorithm [11] and extend the work of Kerzel et al. [9] on a modified DDPG.

2.3 Accelerating Learning and Reward Shaping

One of the major challenges of deep reinforcement learning is the need for extended training periods [10]. Several approaches have been proposed to minimize the learning time: Hafez et al. [7] reward the exploration of novel areas of the state space in a curiosity-driven approach. Schaul et al. [16] implement prioritized memory sweeping [14] to select those samples from the memory that are best suited to drive the learning process.

Reward shaping [15] can guide the learning process by giving small rewards that are, e.g., proportional to the inverse of the distance to a goal. The reward-sparse task of finding a target through random exploration is transformed into the reward-rich task of following the *scent* of the strongest reward, thus accelerating learning. Kerzel et al. [9] suggest a related strategy: instead of creating a reward-rich environment, they augment the virtual size of the target in the simulation according to the difficulty of reaching the target, thus making hard-to-reach targets less difficult to reach. In our experiments, we focus on *reward shaping* and *augmented targets*, as they are representative of learning acceleration strategies that directly simplify the learning task. We evaluate the ability of these strategies especially in the context of task-specific online learning of interactively provided targets.

3 METHODOLOGY

Our proposed architecture for online deep reinforcement learning is composed of two major components, as shown in Figure 2 (left). A modified DDPG learner forms the core of the architecture. The DDPG core is connected via the *mixed-reality interface* to a physical robot, a virtual simulator of the physical robot and *visual inspector*. The robot arm and positions of possible targets, which are both marked with QR code tags, are visually analyzed by the *visual inspector*. The geometry and current configuration, as well as the position of targets, are transferred to the simulator which is used for the trial-and-error stage of deep reinforcement learning. Once the algorithm has learned to reach a given target, the commands are carried out on the real robot.

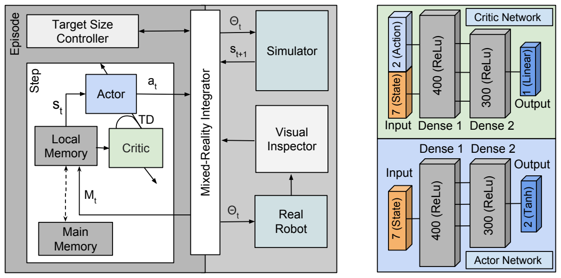

Fig. 2. Left: Architecture for online continuous deep reinforcement learning based on modified DDPG algorithm. Right: Detailed neural architecture of actor and critic.

3.1 Problem Description

Given a target position in 2D, $X^T = [x_1, x_2]$, the joint lengths $L = [l_1, l_2]$ and the initial joint configuration $\Theta_0 = [\theta_0^1, \theta_0^2]$, our goal is to find the final joint configuration $\Theta_t = [\theta_t^1, \theta_t^2]$ where $\|X_{\Theta_t} - X^T\| \leq \rho$ and X_{Θ_t} is the end-effector position corresponding to Θ_t joint configuration at the time step t and ρ is the tolerated error which is the radius of the circle around the object.

3.2 Accelerating Learning with Augmented Targets

One approach to solve this problem(3.1) is to use vanilla DDPG [11]. The agent receives a big reward (here 10) when the end-effector is within the tolerated distance to the target position, and a small reward for every time step (here -0.01).

$$r_t = \begin{cases} 10, & if\ \|X_{\Theta_t} - X^T\| \leq \rho \\ -0.01, & otherwise \end{cases} \tag{5}$$

However, this reward is too sparse to train the agent efficiently. Therefore, we propose an adaptive approach that adjusts the reachability of the target based on the approach by Kerzel et al. [9]. We simplify the task by training a "larger-than-life" target. Keeping the previous reward function, we adjust the augmented target size based on the obtained success during the training. We increase the target size if the agent fails consecutively to reach the target and decrease it when the agent succeeds consecutively to reach the target.

The adaptive target size defined as

$$\tilde{\rho}(e) = \begin{cases} \rho, & if\ e < e_\zeta \\ \tilde{\rho}(e-1) + \delta^+, & if\ \ \eta(10,\ e) < P_\zeta,\ \ e \geq e_\zeta,\ \tilde{\rho}(e) < \tilde{\rho}_{max} \\ \tilde{\rho}(e-1) - \delta^-, & if\ \ \eta(10,\ e) \geq P_\zeta,\ \ e \geq e_\zeta,\ \tilde{\rho}(e) < \tilde{\rho}_{min} \\ \rho, & if\ \ \eta(10,\ e) = 1,\ \ \ e \geq e_\zeta,\ \tilde{\rho}(e) = \rho \\ \tilde{\rho}(e-1), & otherwise \end{cases} \tag{6}$$

where $\tilde{\rho}(e)$ is the augmented target size at episode e, e_ζ is the episode number which the size remaining unchanged (here it is 10). δ^+ and δ^- are the target size increment and decrement values. P_ζ is the success threshold and $\eta(k,\ e)$ indicates the success in reaching the (augmented) target in the past k episodes

$$\eta(k,\ e) = \frac{1}{k} \sum_{e'=e-k}^{e} S_{e'} \tag{7}$$

where S_e is the success/failure of the agent at the end of the episode e (when the episode terminates $t = Tr$).

$$S_e = \begin{cases} 1, if \ \|X_{\Theta_{t=Tr}} - X^T\| \leq \tilde{\rho}(e) \\ 0, if \ \|X_{\Theta_{t=Tr}} - X^T\| > \tilde{\rho}(e) \end{cases} \tag{8}$$

We chose $\delta^+ > \delta^-$ so that the augmented target size grows fast but shrinks slowly to ensure that the difficulty to reach the target does not increase abruptly.

3.3 Accelerating Learning with Reward Shaping

For comparison, we also evaluated a more direct form of reward shaping by giving small rewards based on the change in the distance between the end-effector and the target according to the following formula:

$$rs_t = \frac{\|X_{\Theta_{t-1}} - X^T\| - \|X_{\Theta_t} - X^T\|}{\|X_{\Theta_{t-1}} - X^T\|} \tag{9}$$

where rs_t is the reward calculated by reward shaping at the time step t; however, rs_t is limited to the interval $[-0.05, 0.05]$ to stabilize the rewards.

3.4 Modified DDPG

The agent's experiences (s_t, a_t, r_t, s_{t+1}) are recorded in a memory. The state (s_t) is composed of

$$\begin{aligned} s_t = [&sin(\theta_t^1),\ cos(\theta_t^1),\ sin(\theta_t^2),\ cos(\theta_t^2), \\ &r_t^{target},\ sin(\theta_t^{target}),\ cos(\theta_t^{target})] \end{aligned} \tag{10}$$

where $(r^{target},\ \theta^{target})$ is the polar coordinate of the target position. If the target is augmented, these experiences are also among the recorded samples. Since these experiences are only transient goals to help the agent, we clean the main memory from the augmented target experiences after the agent has learned to reach the object. However, the local memory which is initialized by the main memory at the beginning of each task-specific learning cycle records every experience [9]. In other words, the local memory's content is used for learning to reach one specific target. Therefore, a mixture of novel and old experiences enables learning to reach the new target without catastrophic forgetting [12] regarding the already learned targets.

4 EXPERIMENTS, RESULTS, DISCUSSION

4.1 Robot Platform and Simulator

The robot platform used for our experiments is a 2 DoFs serial manipulator. The joints are rotational servo motors with absolute encoders, and they are specially designed for accurate target angle controlling. The length of each link is adjustable. The setup also provides a ceiling camera to capture top-view images which facilitate the detection process, as depicted in Figure 1. Each target object

Algorithm 1 Modified DDPG Algorithm [11]

```
Initialize critic $\theta^Q$ and actor $\theta^\mu$ networks
Create target networks $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize main and local replay buffer $R_M$, $R_L$
while Agent has NOT learned all Objects do
    for object in Objects do
        obtain the position of the object
        $R_L \leftarrow R_M$
        while Agent has NOT finished the fine tune do
            Receive initial observation state $s_0$
            Define the augmented target size $\tilde{\rho}_e$
            for t = 0, MaxStepNumber do
                Perform action $a_t = \mu(s_t;\theta)$
                if $\tilde{\rho}_e > \rho$ then
                    $R_L \overset{Store}{\leftarrow} (s_t, a_t, r_t, s_{t+1})$
                else
                    $R_L, R_M \overset{Store}{\leftarrow} (s_t, a_t, r_t, s_{t+1})$
                end if
                Update the actor and critic network (eq. 1-4)
            end for
        end while
    end for
end while
```

is associated with a unique QR code The physical specification of the manipulator like the base coordinate frame and the length of each link is calculated and stored to be transferred to the simulator. The link length is calculated by the distance between the centroids of two consecutive joints and end-effector The simulator is the central executing component for learning. The input of the simulator is a relative angle command, and the output is the position of the end-effector. The control frequency is about one KHz and therefore well suited for the requirements of deep reinforcement learning.

4.2 Experimental Design

As shown in Figure 1 (right), five targets are placed in the robot's workspace. In our mixed-reality setup, first, the joint lengths and target positions are extracted through the *Vision Inspector* to construct the simulator robot model.

We evaluate the use of three different learning strategies: in the *baseline* condition, we use an unmodified DDPG algorithm, in the *augmented targets* condition, we adapt the target size dynamically according to learning success as described in Section 3.2, and in the *reward shaping* condition we give direct rewards for just approaching the target as detailed in Section 3.3. Each experimental condition is executed five times to avoid confounding effects or random initialization.

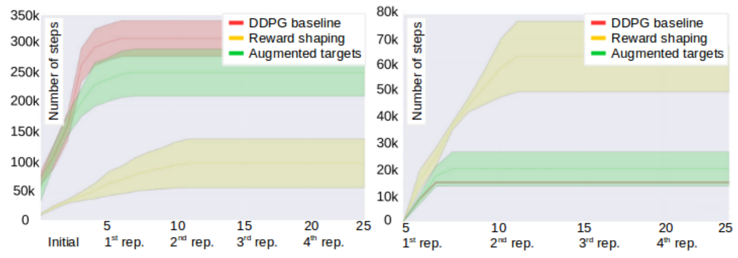

Fig. 3. The accumulated number of training steps to reach the targets T_1 to T_5 five times. Left: The number of all training steps. The X-axis indicates the target number, we train the network to reach each of the five objects repeatedly. The unmodified DDPG baseline requires the most training steps, and the reward shaping condition requires the least. Right: When only accumulating the steps for re-learning already visited targets, the *baseline*, and *augmented targets* condition provide more stable learning.

The robot starts with the first target T_1; in simulation, a solution to reach for the target is learned and then executed on the real robot. The architecture then moves on to learn a solution for the second target T_2 and so on until all five targets have been reached. This is the initial phase, in which each target is novel for the agent. We evaluate the number of (simulated) learning steps it takes in each condition to learn to reach all five targets. In the second phase, the targets are re-visited in the same order four times. This way, we evaluate if learning to reach a target accelerates over the lifetime of the DRL agent. We hypothesize that every time the same target is supplied, the agent learns to reach the object faster.

4.3 Implementation

We implemented our modified DDPG model in Keras [4] with a Tensorflow back end [1]. The hyper-parameters and neural network architecture are adapted from Kerzel et al. [9]. The networks for critic and actor both consist of two dense layers with 400 and 300 units, see Figure 2 (right). We use a low-level input feature vector of the target position in polar space and the current joint configuration in joint space. The outputs are the relative joint values in the range from -1 to 1 from the current value of the motor. For training, we use the Adam optimizer with a learning rate of 10^{-4} for the actor and 2×10^{-4} for the critic. Rectified Linear Unit (ReLU) is used as activation function for all layers except for the last layer of the actor network which produces the joint angles with a hyperbolic tangent function (tanh).

4.4 Results

The results are depicted in Figure 3 (left). As expected, the DDPG *baseline* requires the most training steps to reach all five targets for the first time while the strong *reward shaping* needs the least number of training steps. The *augmented targets* strategy performs slightly better than the *baseline*. To analyze the stability of the learning, we accumulate the number of training steps it takes

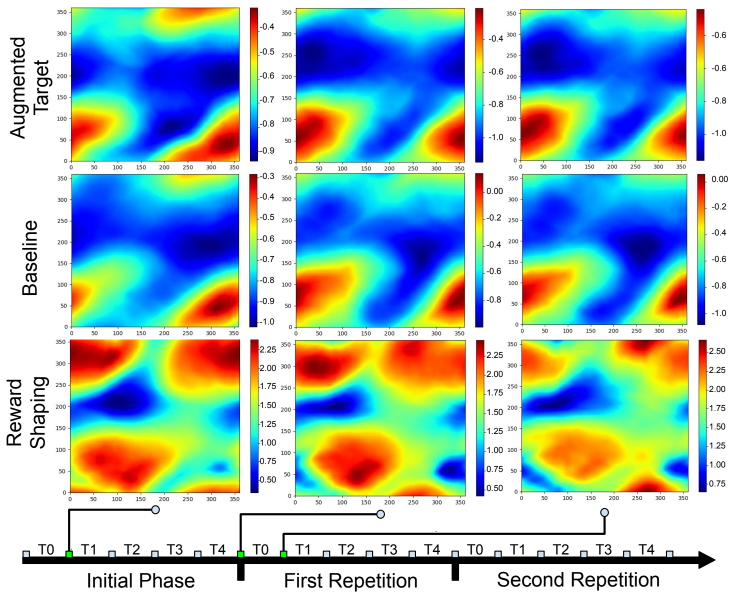

Fig. 4. Visualization of the critic network in joint space. For each experimental condition, the maps show the expected reward for each configuration in the joint space given the target T_1. The rows indicate the training approach and the columns indicate different phases of learning. In each small plot, the X-axis indicates the values of the first joint and the Y-axis indicates the values of the second joint. Regarding the critic value, the *baseline* and *augmented target* approaches are less sensitive compared to *reward shaping* when they are introduced to four new targets.

to re-learn the targets in the repetitions following the initial learning phase, as shown in Figure 3 (right). The *reward shaping* turns out to be less stable than the other two conditions and needs to relearn targets while the *baseline* and *augmented targets* conditions remain mostly stable. We believe the instability in *reward shaping* is caused by the strong reward signal guiding the agent toward the target and satisfying the learning objective immaturely. For further analysis, we take a closer look at the learning process by visualizing the critic network as expected reward maps, see Figure 4. Figure 5 shows a visual representation of an actor map (Policy) for every possible given action to reach a specific target (T_1) in workspace. As our experimental setup uses a 2-DoFs robot arm, it is easy to visualize what the critic learned as a planar map. We focus on the critic network as the actor network is indirectly trained by a gradient transfer from the critic network.

For each experimental condition, we depict the *critic map* for the first and second visit to target T_1. Other targets serve as a distractor to perturb the learning. For stable learning, we expect the initial critic map for the repeated visit of T_1 to be mostly identical to the final map from the first visit. This is not the case for the *reward shaping condition* but true for the other two conditions. Meanwhile in actor map *baseline* and *augmented target* for the first joint are less

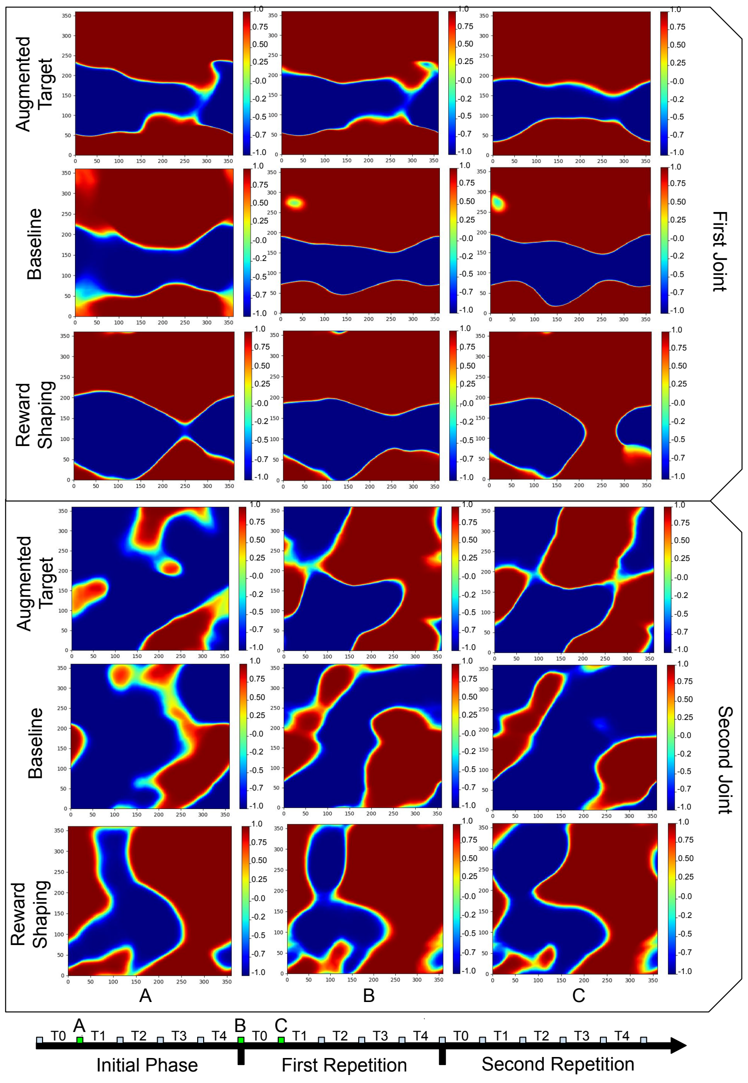

Fig. 5. Visualization of the actor network in joint space. For each experimental condition, the maps show the best action for each state (configuration in the joint space given the targets T_1). The rows indicate the training approach and the columns indicate different phases of learning. In each small plot, the X-axis indicates the values of the first joint and the Y-axis indicates the values of the second joint. The colors in the maps indicate the relative joint values. Similar to critic maps, the actor maps of *baseline* and *augmented target* for the first joint are less sensitive after new targets are introduced. For the second joint, interestingly, learning new targets has improved the action map. This means that despite generating a visually irregular map in the initial phase, the agent has relatively improved the policy.

sensitive after new targets are introduced. For the second joint, learning new targets helped the agent to relatively improve the policy. This means that despite generating a visually irregular map in the initial phase, the agent has relatively improved the policy. Our approach faces the same limitations for transferring learned policies from simulation to the real world based on the accuracy of sensors, actuators and the simulation itself. However, by using the simulation only when requested, we focus the learning process more precisely towards the actions requested by the user.

In summary, the results show that our approach enables stable online learning in a reach-to-grasp task. Different strategies to accelerate the learning can be used depending on the task requirements: learning novel targets can be strongly accelerated by *reward shaping* or more stable learning can be achieved with *augmented targets*. Our approach can be generalized to related deep reinforcement tasks, as long as a suitable way of transferring the state of the physical system into a virtual simulation and a strategy to accelerate learning (by reward shaping, augmenting the targets or another method) are available.

5 Conclusion

We present an approach for online continuous deep reinforcement learning in a mixed-reality environment for a reach-to-grasp task. Based on visual markers, the geometry of the robotic arm and target are transferred to an internal simulation. The approach enables an interactive and secure human-robot collaboration. A non-expert user can operate a physical robot by intuitive instructions via placing a sequence of physical targets. Our modified DDPG algorithm focuses on learning to reach the interactively supplied target; once a solution is found, it is carried out on a real robot.

In contrast to other neural approaches, our online learning approach does not require initial training. Once a target is given, the algorithm focuses on learning to reach it. With each new target, more training samples are generated and used for learning. The approach can use different strategies to speed up the learning process. Depending on priorities, reward shaping can accelerate learning novel targets, or adaptive targets can lead to more stable learning. In future work, we will evaluate dynamic mechanisms to balance strategies for learning acceleration and automated adaptation of hyper-parameters according to the learning progress. Also, the approach will be applied to a more complex robot arm.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
2. Beeson, P., Ames, B.: Trac-ik: An open-source library for improved solving of generic inverse kinematics. In: Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on. pp. 928–935. IEEE (2015). https://doi.org/10.1109/HUMANOIDS.2015.7363472

3. Cangelosi, A., Schlesinger, M.: Developmental Robotics: From Babies to Robots. MIT Press, Cambridge, MA (2015). https://doi.org/10.7551/mitpress/9320.001.0001
4. Chollet, F., et al.: Keras. https://github.com/keras-team/keras (2015)
5. Daya, B., Khawandi, S., Akoum, M.: Applying neural network architecture for inverse kinematics problem in robotics. Journal of Software Engineering and Applications **3**(03), 230 (2010). https://doi.org/10.4236/jsea.2010.33028
6. Gu, S., Lillicrap, T., Sutskever, I., Levine, S.: Continuous deep q-learning with model-based acceleration. In: International Conference on Machine Learning. pp. 2829–2838 (2016), http://dl.acm.org/citation.cfm?id=3045390.3045688
7. Hafez, B., Weber, C., Wermter, S.: Curiosity-driven exploration enhances motor skills of continuous actor-critic learner. In: Proceedings of the 7th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob). pp. 39–46 (2017). https://doi.org/10.1109/DEVLRN.2017.8329785
8. Jha, P., Biswal, B.: A neural network approach for inverse kinematic of a scara manipulator. IAES International Journal of Robotics and Automation **3**(1), 52 (2014). https://doi.org/10.11591/ijra.v3i1.3201
9. Kerzel, M., Beik-Mohammadi, H., Zamani, M.A., Wermter, S.: Accelerating deep continuous reinforcement learning through task simplification (Jul 2018). https://doi.org/10.1109/IJCNN.2018.8489712
10. Levine, S., Pastor, P., Krizhevsky, A., Quillen, D.: Learning hand-eye coordination for robotic grasping with large-scale data collection. In: International Symposium on Experimental Robotics. pp. 173–184. Springer (2016). https://doi.org/10.1177/0278364917710318
11. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
12. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of learning and motivation **24**, 109–165 (1989). https://doi.org/10.1016/S0079-7421(08)60536-8
13. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). https://doi.org/10.1038/nature14236
14. Moore, A.W., Atkeson, C.G.: Prioritized sweeping: Reinforcement learning with less data and less time. Machine learning **13**(1), 103–130 (1993). https://doi.org/10.1007/BF00993104
15. Ng, A.Y., Harada, D., Russell, S.J.: Policy invariance under reward transformations: Theory and application to reward shaping. In: Proceedings of the Sixteenth International Conference on Machine Learning. pp. 278–287. ICML '99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999), http://dl.acm.org/citation.cfm?id=645528.657613
16. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
17. Van Hasselt, H., Wiering, M.A.: Reinforcement learning in continuous action spaces. In: Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on. pp. 272–279. IEEE (2007). https://doi.org/10.1109/ADPRL.2007.368199
18. Watkins, C.J., Dayan, P.: Q-learning. Machine learning **8**(3-4), 279–292 (1992). https://doi.org/10.1007/BF00992698