applied sciences

MDPI

Article

# Motion Planning of Robot Manipulators for a Smoother Path Using a Twin Delayed Deep Deterministic Policy Gradient with Hindsight Experience Replay

**MyeongSeop Kim [1,†], Dong-Ki Han [1,†], Jae-Han Park [2] and Jung-Su Kim [1,*]**

[1] Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea; kimmyungsup57@gmail.com (M.K.); dongki.hann@gmail.com (D.-K.H.)
[2] Robotics R&D Group, Korea Institute of Industrial Technology (KITECH), Ansan 15588, Korea; hans1024@kitech.re.kr
* Correspondence: jungsu@seoultech.ac.kr; Tel.: +82-2-970-6547
† These authors contributed equally to this work.

Received: 2 December 2019; Accepted: 7 January 2020; Published: 13 January 2020


**Abstract:** In order to enhance performance of robot systems in the manufacturing industry, it is essential to develop motion and task planning algorithms. Especially, it is important for the motion plan to be generated automatically in order to deal with various working environments. Although PRM (Probabilistic Roadmap) provides feasible paths when the starting and goal positions of a robot manipulator are given, the path might not be smooth enough, which can lead to inefficient performance of the robot system. This paper proposes a motion planning algorithm for robot manipulators using a twin delayed deep deterministic policy gradient (TD3) which is a reinforcement learning algorithm tailored to MDP with continuous action. Besides, hindsight experience replay (HER) is employed in the TD3 to enhance sample efficiency. Since path planning for a robot manipulator is an MDP (Markov Decision Process) with sparse reward and HER can deal with such a problem, this paper proposes a motion planning algorithm using TD3 with HER. The proposed algorithm is applied to 2-DOF and 3-DOF manipulators and it is shown that the designed paths are smoother and shorter than those designed by PRM.

**Keywords:** motion planning; Probabilistic Roadmap (PRM); Reinforcement learning; policy gradient; Hindsight Experience Replay (HER)

## 1. Introduction

In the Industry 4.0 era, robots and related technologies are fundamental elements of assembley systems in manufacturing; for instance, efficient robot manipulators for various tasks in assembly lines, control of robots with high accuracy, and optimization methods for task scheduling [1,2].

When a task is given from a high level task scheduler, the manipulator has to move its end-effector from the starting point to the goal point without collision with any obstacles or other robots. For this, motion planning algorithms let robot manipulators know how to change their joint angles in order for the end-effector to reach the goal point without collision. Currently, in practice, human experts teach robot manipulators how to move in order to conduct various predefined tasks. Namely, a robot manipulator learns from human experts how to change its joint angles for a given task. However, when the tasks or the working environments change, such manual teaching (a robot manipulator's learning) procedure has to be done again. The other downside of the current approach is optimality or efficiency. In other words, it is not clear if the robot manipulator moves optimally even though the

robot manipulator can perform a given task successfully when a robot manipulator learns from human experts. Therefore, it is important to teach the robot manipulators an optimal path automatically when a task is given. Using policy search-based reinforcement learning, this paper presents a motion planning algorithm for robot manipulators, which makes it possible for the robot manipulator to generate an optimal path automatically; it is a smoother path compared with existing results [3–5].

For robot path planning, sampling-based algorithms find feasible paths for the robot manipulator using a graph consisting of randomly sampled nodes and connected edges in the given configuration space [6,7]. PRM (Probabilistic Roadmaps) and RRT (Rapid Exploring Random Trees) are two representatives of sampling-based planning algorithms. PRM consists of two phases. The learning phase samples nodes randomly from collision-free space in the configuration space and makes edges with direction by connecting the sampled nodes. Then, it constructs a graph using the nodes and edges. The query phase finds the optimal path connecting the starting node and goal node in the graph [8–13]. Note that the resulting path by PRM is made by connecting the sampled nodes in the configuration space; usually, it is not smooth and might be longer than the optimal path. RRT samples nodes from the neighbor of the starting point in the configuration space, constructs a tree by finding a feasible path from the starting node, and expands the graph until the goal point is reached. It works for various environments and can generate a path quickly, but its optimality is not guaranteed in general [14,15]. More recently, Fast Marching Methods (FMMs) using level sets have been proposed for path planning. The FMMs are mainly about efficiently solving the Eikonal equation whose solution provides the optimal path. It is shown that FMMs lead to an asymptotically optimal path and faster convergence than PRM and RRT [16]. Since FMMs, PRM, and RTT are sampling-based approaches, they need a high number of sampling points for high dimensional configuration space in order to obtain a smoother path, which means that they are computationally demanding in calculating the optimal path for given arbitrary starting and ending points. Also, they can suffer from memory deficiency in high dimensional space. However, in the proposed method, when a TD3 agent is trained, the optimal path can easily be computed (i.e., trained neural network computation).

Reinforcement learning is a deep learning approach which finds an optimal policy for an MDP (Markov Decision Process). The agent applies an action according to the policy to the environment and then the agent gets the next state and reward from the environment. The agent finds the optimal policy such that the sum of reward over the horizon is maximized [17,18]. In reinforcement learning, there are two typical approaches to find the optimal policy. Value-based approaches estimate the optimal (action) state value function and derive the corresponding policy from the estimate of the value function [19–21]. On the other hand, policy gradient approaches search the optimal policy directly from the set of state and reward data. It is known that policy gradient approaches show much better performance in general [22–28]. Recently, deep learning-based control and operation of robot manipulators have drawn much attention. In [29,30], robot path planning methods are proposed using a deep $Q$-network algorithm with emphasis on learning efficiency. For path training, a stereo image is used to train DDPG (Deep Deterministic Policy Gradient) in [31]. In [32], a real robot is trained using reinforcement learning for its path planning.

This paper presents a policy gradient-based path planning algorithm. To this end, RAMDP (Robot Arm Markov Decision Process) is defined first. In RAMDP, the state is the joint angle of the robot manipulator and the action is the variation of the joint angle. Then, DDPG (Deep Deterministic Policy Gradient) with HER (Hindsight Experience Replay) is employed for the purpose of searching the optimal policy [24,33]. DDPG is applied since the action in RAMDP is a continuous value and DDPG is devised for MDP with a continuous action. The twin delayed DDPG enhances performance of DDPG so that it shows good convergent property and avoids overestimation. In fact, HER is quite fit to robot path planning since RAMDP is an MDP with sparse reward. Sparse reward means that when an MDP has a finite length of an episode with a specific goal state and the episodes end at non-goal states (say, a failed episode) due to any reasons frequently, the agent can not get much reward. Since all reinforcement learning finds the optimal policy by maximizing the sum of reward,

sparse reward is critical in reinforcement learning. However, as the episodes are saved in the memory, HER modifies the last state in a failed episode as a new goal. Then, the failed episode becomes a normal episode which ends at the goal state. Hence, HER enhances the sample efficiency and fits to the robot path planning. It is shown that such a procedure is quite helpful in a motion planning algorithm. In the proposed algorithm, when the state is computed after applying the action, the collision with obstacle or reaching the goal are checked. It turns out that many states end at non-goal states in the middle of learning. This is why conventional reinforcement learnings do not work well for robot path planning. However, using HER, those episodes can be changed to a normal episode which ends at goal states. Hence, the contribution of the paper is to present a path planning algorithm using DDPG with HER. The proposed method is applied to a 2-DOF and 3-DOF robot manipulators using simulation; experimental results are also shown for a 3-DOF manipulator. In both cases, it is quantitatively demonstrated that the resulting paths are shorter than those by PRM.

## 2. Preliminaries and Problem Setup

### 2.1. Configuration Space and Sampling-Based Path Planning

In sampling-based path planning, configuration space $\mathbf{Q}$ (also called joint space for robot manipulators) represents the space of possible joint angles and is a subset of $n$-dimensional Euclidean space $\mathbb{R}^n$ where $n$ denotes the number of the joints of the manipulator. The values of the joint angles of a robot manipulator are denoted as a point in $\mathbf{Q}$ [1,2,34]. The configuration space consists of two subsets: the collision-free space $\mathbf{Q}_{\text{free}}$ and $\mathbf{Q}_{\text{collide}}$ in which the robot manipulator collides with other obstacles or itself. For motion planning, a discrete representation of the continuous $\mathbf{Q}_{\text{free}}$ is generated by random sampling. Then a connected graph (roadmap) is obtained. The nodes in the graph denote admissible configuration of the robot manipulators, and the edges connecting any two nodes mean feasible paths (trajectory) between the corresponding configurations. Finally, when the starting and goal configurations $q_0$ and $q_{\text{goal}}$ are given, any graph search algorithm is employed to find the shortest path connecting $q_0$ and $q_{\text{goal}}$. There exists a shortest path between $q_0$ and $q_{\text{goal}}$ since they are two nodes on the connected graph.

### 2.2. Reinforcement Learning

Reinforcement learning is an optimization-based method to solve an MDP (Markov Decision Process) [18]. An MDP is comprised of $\{S, A, P, R, \gamma\}$ where $S$ is a set of the state and $A$ denotes a set of the action. Besides, $P$ consists of $P(s'|s, a)$ which is the transition probability that the current state $s \in S$ with the action $a \in A$ becomes the next state $s' \in S$. $R$ stands for the reward function and $\gamma \in [0, 1]$ is the discount factor. The agent's policy $\pi(a|s)$ implies the distribution of the action $a$ for the given state $s$. In reinforcement learning, the agent takes action $a_t$ according to the policy $\pi$ at time $t$ and state $s_t$, and the environment returns the next state $s_{t+1}$ and reward $r_{t+1}$ by the transition $P$ and reward function $R$. By repeating this, the agent updates its policy so as to maximize its expected return $E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}]$. The resulting optimal policy is denoted by $\pi^*$. In order to find the optimal policy, value-based methods like DQN (Deep Q-Network) estimate the optimal value function (i.e., estimate the maximum return) and find the corresponding policy [19]. On the other hand, policy gradient methods compute the optimal policy directly from samples. For instance, REINFORCE, actor-critic method, DPG (Deterministic Policy Gradient), DDPG (Deep DPG), A3C (Asynchronous Advantage Actor-Critic), TRPO (Trust Region Policy Optimization) are representative methods of policy gradient methods [22,35,36]. Training performance of reinforcement learning is heavily dependent on samples which are several sets of the state, action, and next state. Hence, in addition to the various reinforcement learning algorithms, many research efforts have been directed to study on how to use episodes efficiently for the purpose of better agent learning, for example, replay memory [19] and HER (Hindsight Experience Replay) [33]. In this paper, for the sake of designing a motion planning

algorithm, a policy gradient called TD3 (twin delayed Deep Deterministic Policy Gradient) is used for path planning.

## 3. TD3 Based Motion Planning for Smoother Paths

### *3.1. RAMDP (Robot Arm Markov Decision Process) for Path Planning*

In order to develop a reinforcement learning (RL) based path planning, the robot arm MDP (RAMDP) needs to be defined properly first [17]. The state $q_t$ of the RAMDP is the angle value of each joint of the manipulator where the joint angle belongs to the configuration space $\mathbf{Q}$. Hence, for collision-free operation of a robot manipulator, in the motion planning, the state $q_t$ belongs to only $\mathbf{Q}_{\text{free}} \in \mathbb{R}^n$ where $n$ is the number of the joint in the manipulator. In the RAMDP, the action is joint angle variation. Unlike MDP with discrete state and action such as frozen-lake ([17]), the RAMDP under consideration has continuous state (i.e., joint angle) and continuous action (i.e., joint angle variation). Due to this, DDPG or its variants are fit to find the optimal policy for the agent in the RAMDP. In this paper, TD3 (twin delayed DDPG) is employed to find an optimal policy for the continuous action with deterministic policy $\mu$ in the RAMDP. Figure 1 describes how to generate the sample $(q_t,\ a_t,\ q_{t+1},\ r_{t+1})$ in the RAMDP. Suppose that arbitrary initial state $(q_0)$ and goal state $(q_{\text{goal}})$ are given, and that the maximum length of the episode is $T$. At state $q_t$, if the action $a_t$ is applied to the RAMDP, $\hat{q}_{t+1}$ is produced from the RAMDP. Then, according to the reward function in (1), the next state $q_{t+1}$ and $r_{t+1}$ are determined. In view of (1), if $\hat{q}_{t+1}$ does not belong to $\mathbf{Q}_{\text{free}}$, then the next state is set as the current state. Furthermore, if the next state is the goal point, then the reward is 0, and otherwise the reward is $-1$. Then, the whole procedure is repeated until the episode ends. Note that this reward function leads to as short as possible path since TD3 tries to maximize the sum of reward and a longer path implies more $-1$ reward.

$$q_{t+1} = \begin{cases} \hat{q}_{t+1}, & \text{if } \hat{q}_{t+1} \in \mathbf{Q}_{\text{free}} \\ q_t, & \text{if } \hat{q}_{t+1} \notin \mathbf{Q}_{\text{free}} \end{cases}, \qquad r_t = \begin{cases} 0, & \text{if } q_{t+1} == q_{\text{goal}} \\ -1, & \text{if } q_{t+1} \in \mathbf{Q}_{\text{collide}} \\ -1, & \text{if } q_{t+1} \in \mathbf{Q}_{\text{free}} \end{cases} \tag{1}$$

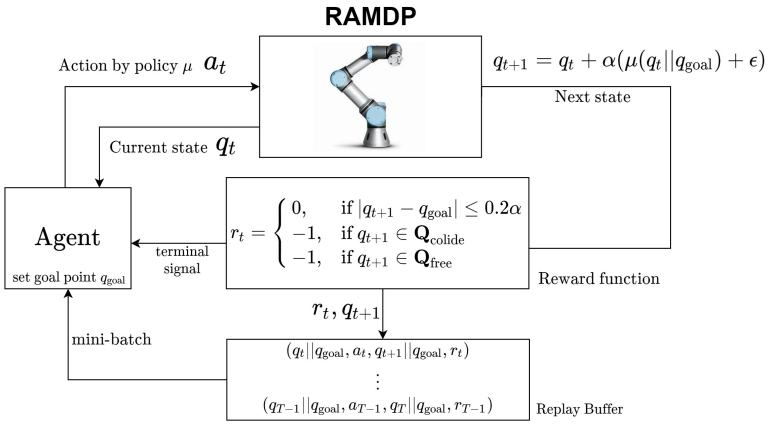



**Figure 1.** Robot Arm Markov Decision Process (RAMDP).

There are two possibilities for the episode to end: (1) the next state becomes the goal state, i.e., $e = [(q_0, a_0), (q_1, r_1, a_1), \cdots, (q_{\text{goal}}, r_g)]$, and (2) the length of the episode is $T$, i.e., $e = [(q_0, a_0), (q_1, r_1, a_1), \cdots, (q_T, r_T)]$ and $q_T \neq q_{\text{goal}}$. For both cases, every sample in the form of $(q_t, a_t, r_t, q_{t+1})$ is saved in the memory. Note that when $\hat{q}_{t+1} \notin \mathbf{Q}_{\text{free}}$, the next state $q_{t+1}$ is set to the current state $q_t$ in order to avoid collision. In the worst case, if this happens repeatedly, the agent is

trained such that the corresponding episode does not occur. Based on these samples in the memory, the optimal policy is determined in accordance with TD3, which is explained in the next subsection.

### *3.2. TD3 (Twin Delayed Deep Deterministic Policy Gradient)*

In this section, TD3 is introduced [24], which is used to search the optimal policy in the proposed algorithm [17,22,35,36]. To this end, it is assumed that a sufficient number of samples $(s_t, a_t, r_t, s_{t+1})$ are stored in the memory.

Figure 2 describes the structure of TD3. The basic structure of TD3 is the actor-critic network. However, unlike the actor critic network, there are two critic deep neural networks (DNN), one actor DNN, and three target DNNs for each critic and actor DNNs. Hence, there are 6 DNNs (2 critic DNNs and their target DNNs, 1 actor DNN, and its target DNN) in TD3. As seen in Figure 3, the critic DNNs generate an optimal estimate $Q_{\theta_i}(s_t, a_t)$ of the state-action value function. The input of the critic DNN is a mini-batch from the memory and its output is $Q_{\theta_i}(s_t, a_t)$, $i = 1, 2$. The mini-batch is a finite set of samples $(s_t, a_t, r_t, s_{t+1})$ from the memory. In TD3, it is important that the two critic DNNs are used in order to remove overestimation bias in $Q_{\theta_i}(s_t, a_t)$ function approximation. The overestimation bias can take place when bad states due to accumulated noises are overestimated. In order to cope with this, TD3 chooses the smaller $Q_{\theta_i}(s_t, a_t)$ value out of two critic DNN outputs critic DNN as the target value. In Figure 3, $\theta_1$ and $\theta_2$ are the parameters of the two critic DNNs, and $\theta_1'$ and $\theta_2'$ those of corresponding target DNNs. In order to train the critic DNN, as the cost function, the following quadratic function of temporal difference error $\delta := Q_\theta(q, a) - y^{\text{target}}$ is minimized,

$$J_\delta(\theta) := 1/2(Q_\theta(q, a) - y^{\text{target}})^2, \tag{2}$$

where $Q_\theta(s, a)$ stands for the parameterized state-action value function $Q$ with parameter $\theta$,

$$y^{\text{target}} = r + \gamma \min_{i=1,2} Q_{\theta_i'}(q, \tilde{a}) \tag{3}$$

is the target value of the function $Q_\theta(s, a)$, and the target action (i.e., the action used in the critic target DNN) is defined as,

$$\tilde{a} = \mu_\phi(q) + \bar{\epsilon}, \tag{4}$$

where noise $\bar{\epsilon}$ follows a clipped normal distribution $\text{clip}[(\mathcal{N}(0, \sigma), -c, c],\ c > 0$. This implies that $\bar{\epsilon}$ is a random variable with $\mathcal{N}(0, \sigma)$ and belongs to the interval $[-c, c]$.

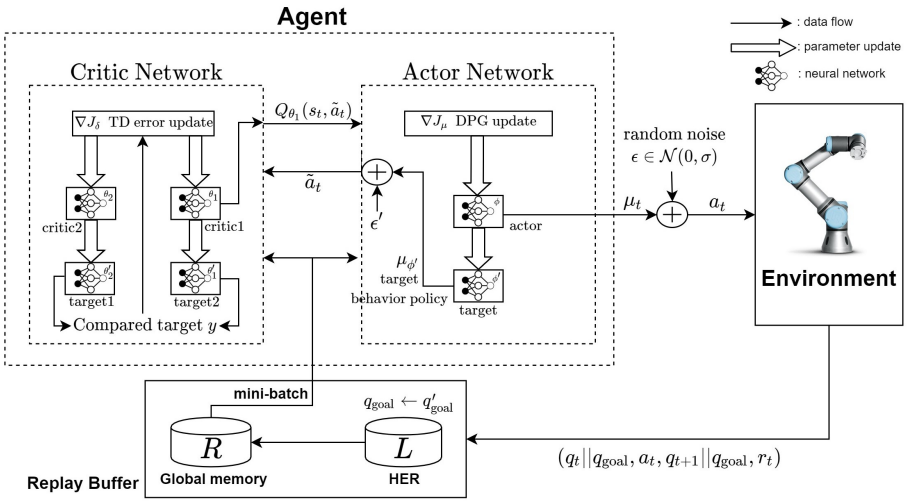



**Figure 2.** Structure of TD3 (Twin Delayed Deep Deterministic Policy Gradient) with RAMDP.

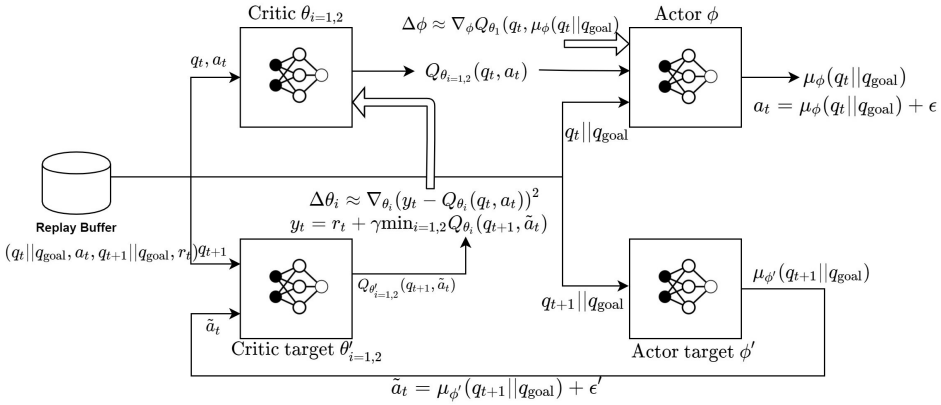



**Figure 3.** Details of critic and actor deep neural network for RAMDP.

The inputs of the actor DNN are both $Q_{\theta_1}(q_t, a_t)$ from the critic DNN and the mini-batch from the memory, and the output is the action. Precisely, the action is given by $a_t = \mu_\phi(q_t) + \epsilon$ where $\phi$ is the parameter of the actor DNN, $\mu_\phi$ is the output from the actor DNN, and a deterministic and continuous value. Noise $\epsilon$ follows the normal distribution $\mathcal{N}(0, \sigma)$, and is added for exploration. In order to tune the parameter $\phi$, the following cost function is minimized.

$$J_\mu(\phi) = \sum_q d^\mu(q) Q_\theta(q, \mu_\phi(q) + \epsilon), \tag{5}$$

where $d^\mu(q)$ denotes the distribution of the state. Note that the gradient $\nabla J_\mu(\theta)$ (equivalently $\nabla Q_\theta(q, a)$) is used to update the parameter $\phi$. This is why the method is called the policy gradient.

Between the two outputs from the two critic target DNN, the common target value in (3) is used to update the two critic DNNs. Also, TD3 updates the actor DNN and all three target DNNs every $d$ steps periodically in order to avoid too fast convergence. Note that the policy $\mu_\phi(q)$ is updated proportionally to only $Q_{\theta_1}(q, \mu_\phi(q))$ [24]. The parameters for the critic target DNN and the actor target DNN are updated according to $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$ at every $d$ steps, which not only maintain small temporal difference error, but also make the parameters in the target DNN updated slowly.

### *3.3. Hindsight Experience Replay*

Since the agent in reinforcement learning is trained from samples, it is utmost important to have helpful samples which mean that the action state pair improve the action value function. On the other hand, many unhelpful samples are generated in RAMDP since many episodes end without reaching the goal. In other words, RAMDP is an MDP with sparse reward. For the purpose of enhancing sample efficiency, HER (Hindsight Experience Replay) is employed in this paper. For the episode $e = [(q_0, a_0), (q_1, r_1, a_1), \cdots, (q_T, r_T)]$ in the memory where $q_T$ is not the goal state, HER resets $q_T$ as $q_{\text{goal}}$. This means that even though the episode does not end the goal state, the episode becomes a ended state at the goal after modification by HER. Then, the failed episodes can be use to train the agent since the modified episodes are a goal achieved episodes. Algorithm 1 summarizes the proposed algorithm introduced in this section.

**Algorithm 1** Training procedure for motion planning by TD3 with HER. The red part is for a robot manipulator, and the blue part is for HER.

Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$, and actor network $\pi_\phi$ with $\theta_1, \theta_2, \phi$
Initialize target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{R}$
**for** $e = 1$ `to` $M$ **do**
    Initialize local buffer $\mathcal{L}$ ▷ memory for HER
    **for** $t = 0$ `to` $T-1$ **do**
        Randomly choose goal point $q_{\text{goal}} \in \mathbf{Q}_{\text{free}}$
        Select action with noise:
        $a_t \sim \mu_\phi(q_t||q_{\text{goal}}) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma)$ ▷ $||$ denotes concatenation
        $q_{t+1} = q_t + \alpha * a_t$

        **if** $q_t + \alpha(\mu_\phi(q_t||q_{\text{goal}}) + \epsilon) \notin \mathbf{Q}$ **then**
            $q_{t+1} \leftarrow q_t$
        **else if** $q_{t+1} \in \mathbf{Q}_{\text{collide}}$ **then**
            $q_{t+1} \leftarrow q_t$
        **else if** $|q_{t+1} - q_{\text{goal}}| \leq 0.2 * \alpha$ **then**
            Terminal by goal arrival
        **end if**

        $r_t := r(q_t, a_t, q_{\text{goal}})$
        Store the transition $(q_t||q_{\text{goal}}, a_t, r_t, q_{t+1}||q_{\text{goal}})$ in $\mathcal{R}, \mathcal{L}$

        Sample mini-batch of $n$ transitions $(q_j||q_{\text{goal}}, a_j, r_j, q_{j+1}||q_{\text{goal}})$ from $\mathcal{R}$
        $\tilde{a}_j \leftarrow \mu_{\phi'}(q_{j+1}||q_{\text{goal}}) + \epsilon,\ \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma),\ c, -c)$
        $y_j \leftarrow r_j + \gamma \min_{i=1,2} Q_{\theta_i'}(q_{j+1}, \tilde{a}_j)$
        Update critics $\theta_i$ with temporal difference error:
        $\nabla_{\theta_i} J(\theta_i) = \frac{1}{n} \sum_{j=1}^{n} \nabla_{\theta_i} (y_j - Q_{\theta_i}(q_j, a_j))^2$

        **if** $t$ mod $d$ **then** ▷ delayed update with $d$
            Update actor $\phi$ by the deterministic policy gradient:
            $\nabla_\phi J(\phi) = \frac{1}{n} \sum_{j=1}^{n} \nabla_\phi Q_{\theta_1}(q_j, \mu_\phi(q_j||q_{\text{goal}}))$
            Update target networks:
            $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
            $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
        **end if**
    **end for**
    **if** $q_T \neq q_{\text{goal}}$ **then**
        Set additional goal $q'_{\text{goal}} \leftarrow q_T$
        **for** $t = 0$ `to` $T-1$ **do**
            Sample a transition $(q_t||q_{\text{goal}}, a_t, r_t, q_{t+1}||q_{\text{goal}})$ from $\mathcal{L}$
            $r'_t := r(q_t, a_t, q'_{\text{goal}})$
            Store the transition $(q_t||q'_{\text{goal}}, a_t, r_t, q_{t+1}||q'_{\text{goal}})$ in $\mathcal{R}$
        **end for**
    **end if**
**end for**

## 4. Case Study for 2-DOF and 3-DOF Manipulators

In order to show the effectiveness of the proposed method, it is applied to robot manipulators. Table 1 shows the information of the used 2-DOF and 3-DOF and manipulators.

**Table 1.** Parameters of 2-DOF and 3-DOF manipulators.

| DOF | Joint Max (degree) | Joint Min (degree) | Action Step Size ($\alpha$) | Goal Boundary |
|---|---|---|---|---|
| 2 | (140, −45, 150) | (−140, −180, 45) | 0.1381 | 0.2 |
| 3 | (60, 60) | (0, 0) | 3.0 | 1.0 |

For easy visualization, the proposed algorithm is applied to the 2-DOF manipulator first. Table 2 summarizes the tuning parameters for the designed TD3 with HER.

**Table 2.** Tuning parameters for the designed TD3 with HER.

| Network Name | Learning Rate | Optimizer | Update Delay | DNN Size |
|---|---|---|---|---|
| Actor | 0.001 | adam | 2 | $6 \times 400 \times 300 \times 3$ |
| Critic | 0.001 | adam | 0 | $6 \times 400 \times 300 \times 1$ |
| Actor target | 0.005 | - | 2 | $6 \times 400 \times 300 \times 3$ |
| Critic target | 0.005 | - | 2 | $6 \times 400 \times 300 \times 1$ |

In order to train TD3 with HER for the 2-DOF manipulator, 8100 episodes are used. Figure 4 describes the success ratio of each episode with HER when the network is training. In other words, when the network is learning with arbitrary starting and goal points, sometimes the episodes end at the given goal point but sometimes the episodes end before reaching the given goal point. In Figure 4, the green lines denote the success ratio of every 10 episodes and the thick lines stand for the moving average of the gray lines. Figure 5 shows the reward as the number of the episode increases, i.e., the training is proceeding. The reward converges as the number of the episode increases. In view of the results in Figures 4 and 5, we can see that learning is over successfully.

For the purpose of testing the trained TD3, it is verified if the optimal paths are generated when random starting and goal points are given to the trained TD3. For testing, only the actor DNN without its target DNN is used with the input being a starting and goal point repeatedly. The input to the trained actor DNN is $(q_t, q_{\text{goal}})$, the output is $(a_t, q_{t+1})$, and then this is repeated with $q_t =: q_{t+1}$ as depicted in Figure 6.

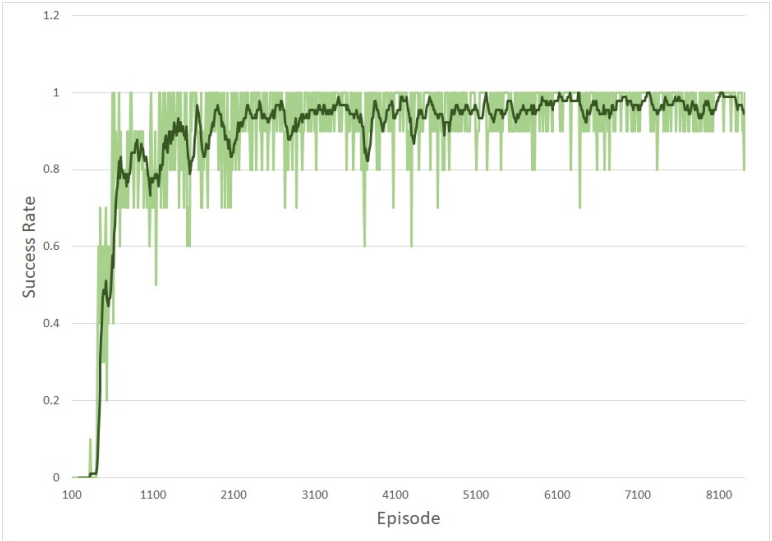



**Figure 4.** 2-DOF manipulator: success ration of reaching the goal point with HER.

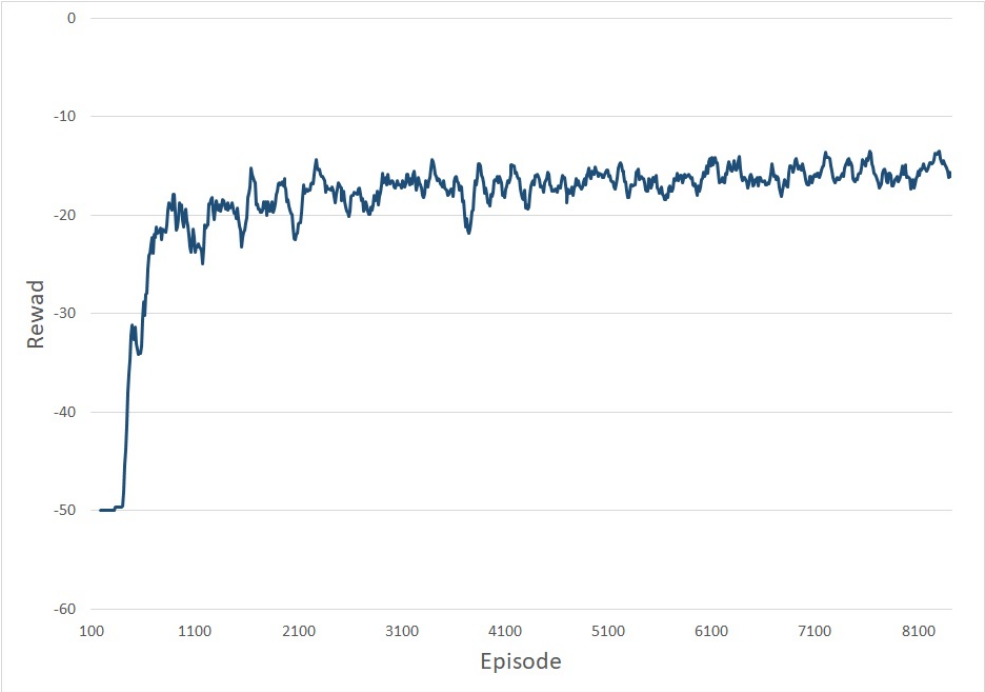



**Figure 5.** 2-DOF manipulator: reward from learning.

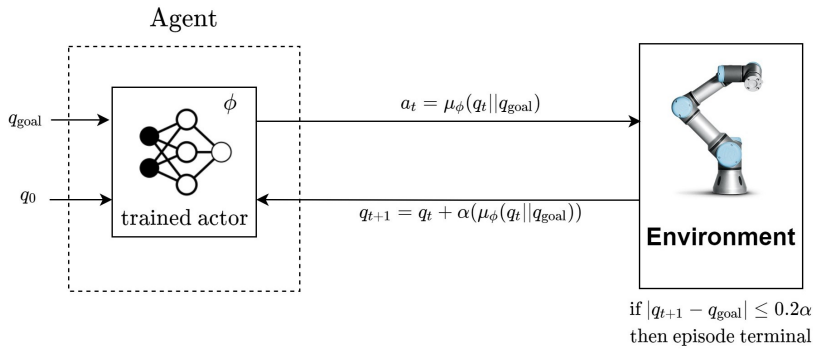



**Figure 6.** Path generation using the trained actor DNN.

When this is applied to the 2-DOF manipulator, the resulting paths are shown in Figure 7. In Figure 7, the green areas represent obstacles in the configuration space, and the rhombus denote the starting point and the circles means the goal point. For comparison, PRM is also applied to generate the paths with the same starting and goal points. As shown in Figure 7, the proposed method generates smoother paths in general. This is confirmed from many other testing data as well. In Figure 7, the red lines are the resulting paths by PRM and the blue lines by the proposed method. In average, the resulting path by the proposed method is shortened by 3.45% compared with the path by PRM.

In order to test the proposed method for a real robot manipulator, the 3-DOF open manipulator. For details, see http://en.robotis.com/model/page.php?co_id=prd_openmanipulator is considered. For easy understanding, Figure 8 shows the workspaces in Matlab and Gazebo in ROS (Robot Operating System) respectively, and configuration space of the open manipulator with four arbitrary obstacles.

The tuning parameters for the TD3 with HER are also shown in Table 2.

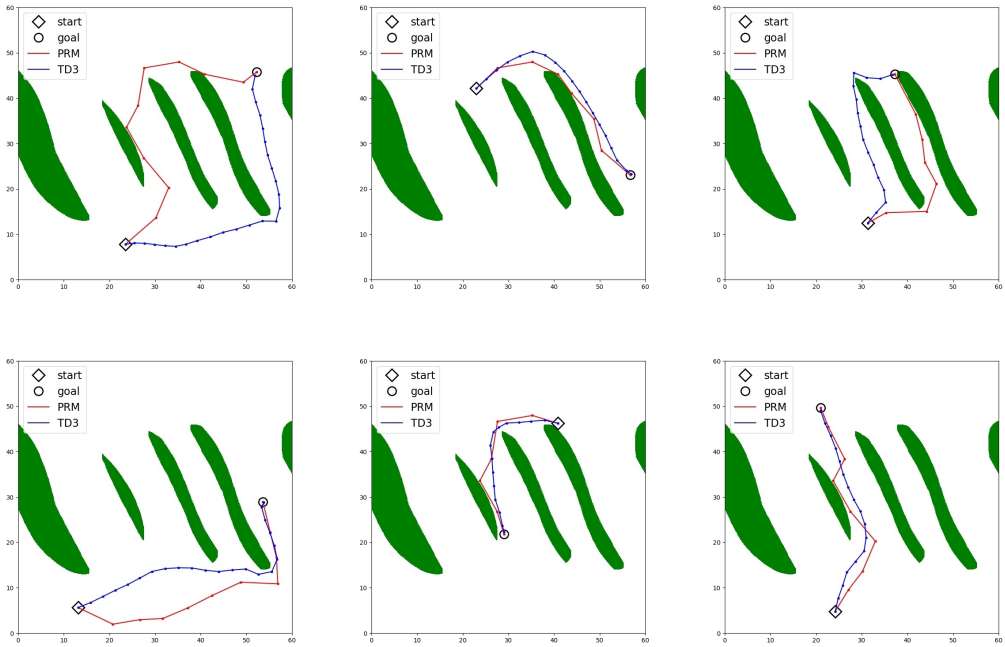
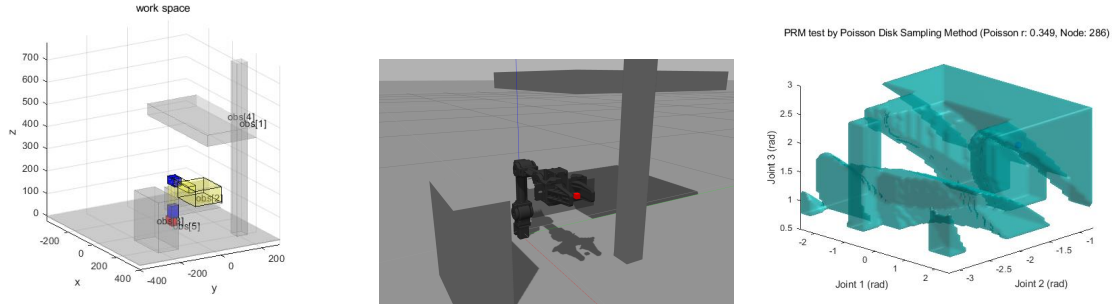



**Figure 7.** DDPG based path generation for arbitrary starting and goal points in C-space.




(**a**) Workspace in Matlab (**b**) Workspace in Gazebo (in ROS) (**c**) Configuration space

**Figure 8.** Workspace and configuration space. (**a**) Workspace in Matlab; (**b**) Workspace in Gazebo (in ROS); (**c**) Configuration space.

For training, 140000 episodes are used. In view of the converged reward and success ratio in Figures 9 and 10, we can see that the learning is also over successfully. With this result, random starting and goal points are given to the trained network in order to obtain a feasible path between them. Figure 11 shows several examples of generated paths by the trained actor DNN when arbitrary starting points and goal points are given. The red lines are resulting paths by PRM and the blue lines by the proposed method. As seen in the figure, the proposed method results in smoother and shorter paths in general. For the sake of between comparison, 100 arbitrary starting and gold points are used to generate paths using PRM and the proposed method. Figure 12 shows the lengths of the resulting 100 paths. In light of Figure 12, it is obvious that the proposed method generates smoother and shorter paths in general. Note that, in average, the resulting path by the proposed method is shortened by 2.69% compared with the path by PRM.

When the proposed method is also implemented to the open manipulator, the same experimental result as the simulation was obtained. The experimental result is presented in the https://sites.google.com/site/cdslweb/publication/td2-demo.

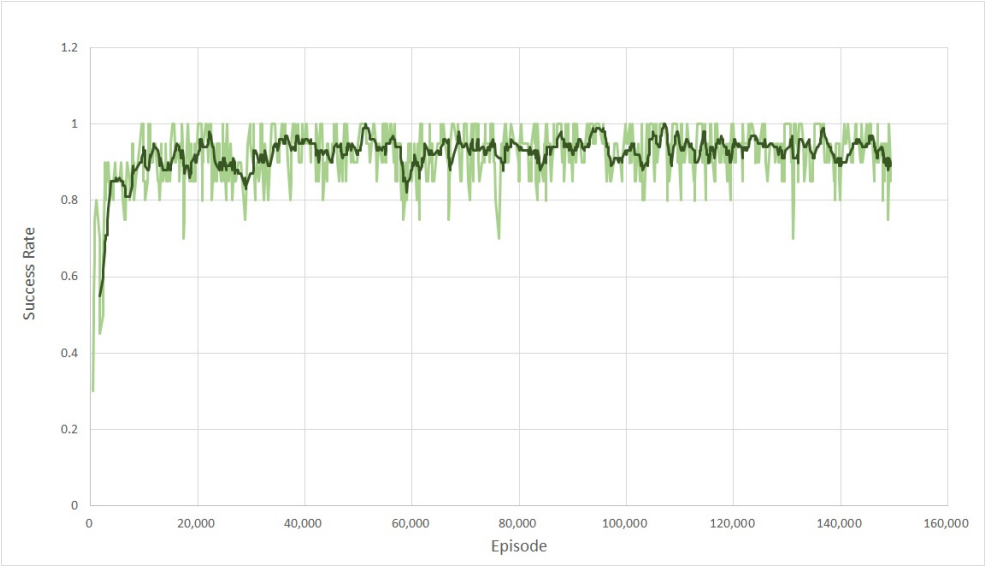



**Figure 9.** 3-DOF manipulator: success ration of reaching the goal point with HER.

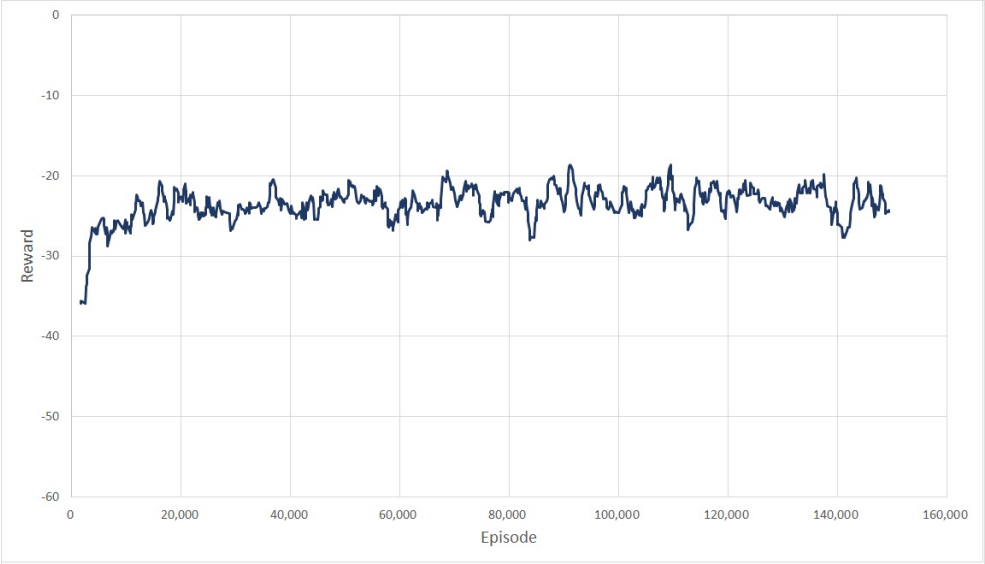



**Figure 10.** 3-DOF manipulator: reward from learning.

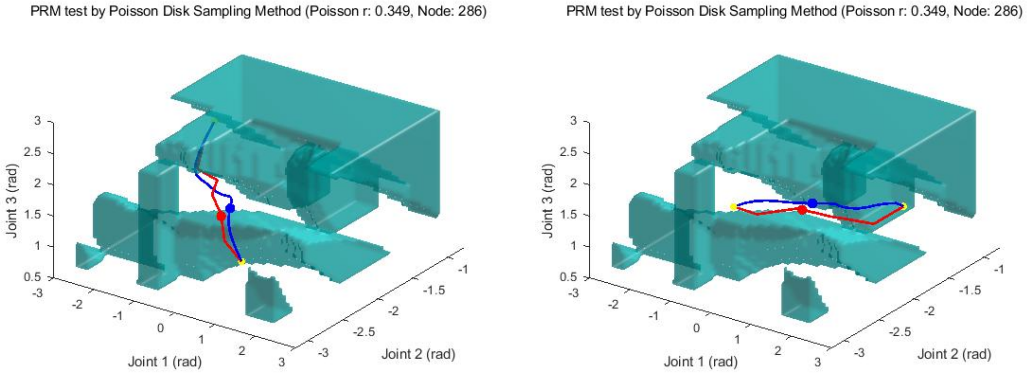



**Figure 11.** *Cont*.

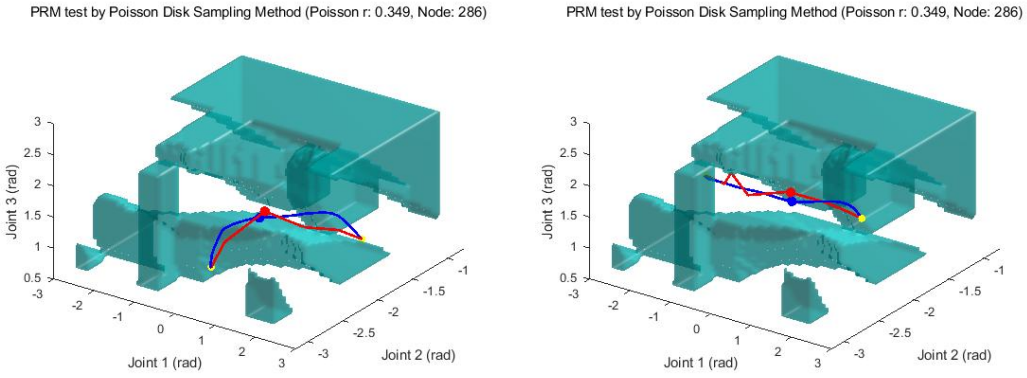



**Figure 11.** Path generation using DDPG with HER.

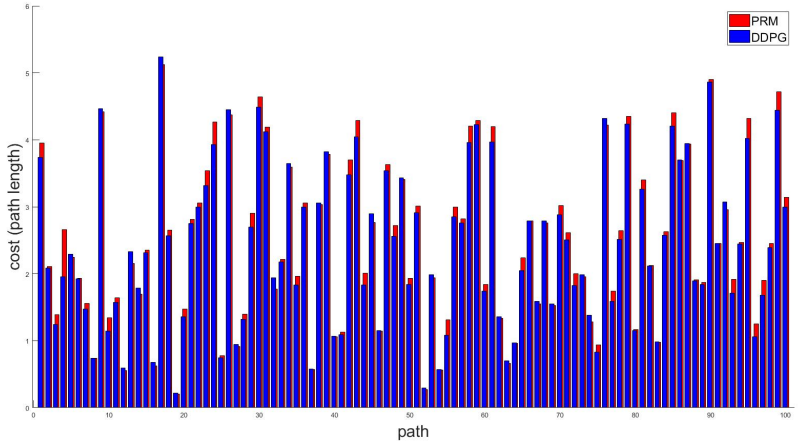



**Figure 12.** Comparison of paths by PRM and the proposed method.

## 5. Conclusions

For the purpose of enhancing efficiency in manufacturing industry, it is important to improve performance of robot path planning and tasking scheduling. This paper presents a reinforcement learning-based motion planning method of robot manipulators with focus on smoother and shorter path generation, which means better operation efficiency. To this end, motion planning problem is reformulated as a MDP (Markov Decision Process), called RAMDP (Robot Arm MDP). Then, TD3 (twin delayed deep deterministic policy gradient, twin delayed DDPG) with HER (Hindsight Experience Replay) is designed. DDPG is used since the action in RAMDP is a continuous value and DDPG is a policy gradient tailored to an MDP with a continuous action. Moreover, since many failed episodes are generated in the RAMDP meaning that the episode ends at non-goal state due to mainly collision, HER is employed in order to enhance sample efficiency.

Future research topic includes how to solve motion planning problem for multi-robot arms whose common working space is non-empty. To solve this problem, configuration space augmentation might be a candidate solution. Since the augmented configuration space becomes high dimensional, it would be interesting to compare performance of the proposed reinforcement learning-based approach by that of sampling-based approaches such as FMMs, PRM, and RTT. Moreover, reinforcement learning-based motion planning for dynamic environment is also a challenge problem.

**Author Contributions:** M.K. and D.-K.H. surveyed the backgrounds of this research, designed the preprocessing data, designed the deep learning network, and performed the simulations and experiments to show the benefits of the proposed method. J.-S.K. and J.-H.P. supervised and supported this study. All authors have read and agreed to the published version of the manuscript.

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

**Acknowledgments:** This work was supported by the Technology Innovation Program (or Industrial Strategic Technology Development Program) (10080636, Development of AI-based CPS technology for Industrial robot applications) funded By the Ministry of Trade, Industry & Energy(MOTIE, Korea), and by the Human Resources Development of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Ministry of Trade, Industry & Energy of the Korea government (No. 20154030200720).

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

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| MDP | Markov Decision Process |
| RAMDP | Robot Arm Markov Decision Process |
| DOF | Degrees Of Freedom |
| PRM | Probabilistic Roadmaps |
| RRT | Rapid Exploring Random Trees |
| FMMs | Fast Marching Methods |
| DNN | Deep Neural Networks |
| DQN | Deep Q-Network |
| (A3C) | Asynchronous Advantage Actor-Critic |
| (TRPO) | Trust Region Policy Optimization |
| (DPG) | Deterministic Policy Gradient |
| DDPG | Deep Deterministic Policy Gradient |
| TD3 | Twin Delayed Deep Deterministic Policy Gradient |
| HER | Hindsight Experience Replay |
| (ROS) | Robot Operating System |

## References

1. Laumond, J.P. *Robot Motion Planning and Control*; Springer: Berlin, Germany, 1998; Volume 229.
2. Choset, H.M.; Hutchinson, S.; Lynch, K.M.; Kantor, G.; Burgard, W.; Kavraki, L.E.; Thrun, S. *Principles of Robot Motion: Theory, Algorithms, and Implementation*; MIT Press: Cambridge, MA, USA, 2005.
3. Cao, B.; Doods, G.; Irwin, G.W. Time-optimal and smooth constrained path planning for robot manipulators. In Proceedings of the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA, USA, 8–13 May 1994; pp. 1853–1858.
4. Kanayama, Y.J.; Hartman, B.I. Smooth local-path planning for autonomous vehicles1. *Int. J. Robot. Res.* **1997**, *16*, 263–284. [CrossRef]
5. Rufli, M.; Ferguson, D.; Siegwart, R. Smooth path planning in constrained environments. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3780–3785.
6. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
7. Spong, M.W.; Hutchinson, S.A.; Vidyasagar, M. Robot modeling and control. *IEEE Control Syst.* **2006**, *26*, 113–115.
8. Kavraki, L.E.; Kolountzakis, M.N.; Latombe, J.C. Analysis of probabilistic roadmaps for path planning. *IEEE Trans. Robot. Autom.* **1998**, *14*, 166–171. [CrossRef]
9. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
10. Kavraki, L.E. Random Networks in Configuration Space for Fast Path Planning. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 1995.
11. Kavraki, L.E.; Latombe, J.C.; Motwani, R.; Raghavan, P. Randomized query processing in robot path planning. *J. Comput. Syst. Sci.* **1998**, *57*, 50–60. [CrossRef]
12. Bohlin, R.; Kavraki, L.E. A Randomized Approach to Robot Path Planning Based on Lazy Evaluation. *Comb. Optim.* **2001**, *9*, 221–249.

13. Hsu, D.; Latombe, J.C.; Kurniawati, H. On the probabilistic foundations of probabilistic roadmap planning. *Int. J. Robot. Res.* **2006**, *25*, 627–643. [CrossRef]
14. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; Volume 2, pp. 995–1001.
15. Lavalle, S.; Kuffner, J. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*; CRC Press: Boca Ratol, FL, USA, 2000; pp. 293–308.
16. Janson, L.; Schmerling, E.; Clark, A.; Pavone, M. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *Int. J. Robot. Res.* **2015**, *34*, 883–921. [CrossRef] [PubMed]
17. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2011.
18. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed.; John Wiley & Sons, Inc.: New York, NY, USA, 1994.
19. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.; Veness, J.; Bellemare, M.; Graves, A.; Riedmiller, M.; Fidjeland, A.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
20. Langford, J. Efficient Exploration in Reinforcement Learning. In *Encyclopedia of Machine Learning and Data Mining*; Sammut, C., Webb, G.I., Eds.; Springer US: Boston, MA, USA, 2017; pp. 389–392.
21. Tokic, M. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *Annual Conference on Artificial Intelligence*; Springer: Berlin, Germany, 2010; pp. 203–210.
22. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.M.O.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
23. Hasselt, H.v.; Guez, A.; Silver, D. Deep Reinforcement Learning with Double Q-Learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 2094–2100.
24. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. *arXiv* **2018**, arXiv:1802.09477.
25. Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
26. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; pp. 1928–1937.
27. Degris, T.; Pilarski, P.M.; Sutton, R.S. Model-free reinforcement learning with continuous action in practice. In Proceedings of the 2012 American Control Conference (ACC), Montreal, QC, Canada, 27–29 June 2012; pp. 2177–2182.
28. Degris, T.; White, M.; Sutton, R.S. Off-policy actor-critic. *arXiv* **2012**, arXiv:1205.4839.
29. Bae, H.; Kim, G.; Kim, J.; Qian, D.; Lee, S. Multi-Robot Path Planning Method Using Reinforcement Learning. *Appl. Sci.* **2019**, *9*, 3057. [CrossRef]
30. Lv, L.; Zhang, S.; Ding, D.; Wang, Y. Path Planning via an Improved DQN-Based Learning Policy. *IEEE Access* **2019**, *7*, 67319–67330. [CrossRef]
31. Paul, S.; Vig, L. Deterministic policy gradient based robotic path planning with continuous action spaces. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 725–733.
32. Gu, S.; Holly, E.; Lillicrap, T.; Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3389–3396.
33. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, O.P.; Zaremba, W. Hindsight experience replay. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5048–5058.
34. Lozano-Pérez, T. Spatial Planning: A Configuration Space Approach. In *Autonomous Robot Vehicles*; Cox, I.J.; Wilfong, G.T., Eds.; Springer New York: New York, NY, USA, 1990; pp. 259–271.

35. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic Policy Gradient Algorithms. In Proceedings of the 31st International Conference on International Conference on Machine Learning, Bejing, China, 22–24 June 2014; pp. I-387–I-395.
36. Sutton, R.S.; Mcallester, D.; Singh, S.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*; MIT Press: Cambridge, MA, USA, 2000; pp. 1057–1063.