sensors

Article

Path Planning for Multi-Arm Manipulators Using Deep Reinforcement Learning: Soft Actor–Critic with Hindsight Experience Replay

Evan Prianto [1,†], MyeongSeop Kim [1,†], Jae-Han Park [2], Ji-Hun Bae [2] and Jung-Su Kim [1,*]

1 Research Center for Electrical and Information Technology, Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea; evanprianto.el@gmail.com (E.P.); kimmyungsup57@gmail.com (M.K.)
2 Applied Robot R&D Department, Korea Institute of Industrial Technology (KITECH), Ansan 15588, Korea; hans1024@kitech.re.kr (J.-H.P.); joseph@kitech.re.kr (J.-H.B.)
* Correspondence: jungsu@seoultech.ac.kr; Tel.: +82-2-970-6547
† These authors contributed equally to this work.

Received: 11 September 2020; Accepted: 17 October 2020; Published: 19 October 2020

Abstract: Since path planning for multi-arm manipulators is a complicated high-dimensional problem, effective and fast path generation is not easy for the arbitrarily given start and goal locations of the end effector. Especially, when it comes to deep reinforcement learning-based path planning, high-dimensionality makes it difficult for existing reinforcement learning-based methods to have efficient exploration which is crucial for successful training. The recently proposed soft actor–critic (SAC) is well known to have good exploration ability due to the use of the entropy term in the objective function. Motivated by this, in this paper, a SAC-based path planning algorithm is proposed. The hindsight experience replay (HER) is also employed for sample efficiency and configuration space augmentation is used in order to deal with complicated configuration space of the multi-arms. To show the effectiveness of the proposed algorithm, both simulation and experiment results are given. By comparing with existing results, it is demonstrated that the proposed method outperforms the existing results.

Keywords: path planning; multi-arm manipulators; reinforcement learning; soft actor-critic (SAC); hindsight experience replay (HER); collision avoidance

1. Introduction

In the Industry 4.0 era, one of the important elements in the manufacturing industry for a smart factory is automation via collaboration of robot manipulators, and the manufacturing industry has been less affected by human workforce [1]. Especially, multi-arm manipulators have been drawing more attention due to its increasing application such as assembly line in a factory, space research, customer service, exploration, even rescue mission [2]. Hence, it is utmost important to improve the efficiency of the operation of the multi-arm manipulator.

1.1. Motivation

For manipulators in manufacturing industry, when a specific task (e.g., moving an object) is given, human experts find manually a collision-free path from the start and goal locations for the end-effector in order to perform the task and teach the path to the manipulator. Hence, if the start and goal locations of the end-effector vary owing to task change, this procedure has to be repeated. Moreover, in the case of multi-arm manipulators, such a procedure becomes much more difficult [3,4]. Hence, it is important to make such a procedure carried out automatically. In addition to this, for the purpose of making the whole manufacturing process efficient and optimal, the multi-arm manipulator has to learn the optimal (i.e., shortest) path rather than a feasible path when arbitrary start and goal locations are given [5,6].

1.2. Background and Related Work

A representative of path planning for multi-arm manipulators is the sampling-based algorithm which computes the path after building a graph using sampled points of the workspace [7]. The examples of sampling-based algorithms are fast marching method (FMM) [8], probabilistic road map (PRM) method [9], and rapid exploring random trees (RRT) method [10–13]. In PRM, the Dijkstra algorithm is applied to find the shortest paths from the constructed graph [14,15]. The path generated by the sampling-based method may not be the optimal path since the resulting path is heavily dependent on the sampling methods. The other approach to solve the path planning problem for multi-arm manipulators is to use the artificial potential function to create a motion equation that can guide the robot arm to the goal point [16,17]. By computing the gradient of the equation, the direction of the optimal path can be attained [18]. However, it can be trapped in the local minimum of the potential field and fail to find the right path [19]. This situation can be worse if the path planning problem is high-dimensional. Note that the path planning under consideration is complicated and high-dimensional by nature. Due to this reason, an effective path planning algorithm for multi-arm manipulators has to be developed. In the literature about path planning, there are already some deep learning-based approaches implemented for robot applications such as mobile manipulation [20,21], unmanned ship [22] and even for multi-mobile robot [23]. These imply that deep learning-based approach can be promising in path planning for single arm manipulator [24] and also for multi-arm manipulators [25]. Especially, the focus of this paper is placed on devising a deep reinforcement learning-based path planning algorithm for multi-arm manipulators [26].

There are mainly three difficulties in reinforcement learning-based path planning for the multi-arm manipulator. First, in order to design a path planning algorithm, although it is essential to deal with the configuration space properly, it is not easy to define the configuration space which consists of collision-free space and collision space for the multi-arm manipulator since there are multiple arms, which makes the problem nontrivial. Second, when reinforcement learning-based path planning is designed, efficient exploration is indispensable to find the shortest path. However, since path planning for the multi-arm manipulator is a high-dimensional problem, existing reinforcement learning-based path planning suffers from poor exploration performance, which makes the resulting path planning lead to a suboptimal solution (i.e., not the shortest path). Third, when path planning is designed using reinforcement learning, the agent can get the reward in a sparse manner. In other words, when arbitrary start and goal locations are given for the reinforcement learning-based path planning, there can be many cases where the agent fails to find the shortest path due to physical limitations or high dimensionality of the configuration space. In this case, the agent can not get the reward contributing to training.

1.3. Proposed Solution

In this paper, a recently reported policy gradient type reinforcement learning algorithm called SAC (Soft Actor-Critic)-based path planning is proposed to overcome the previously described difficulties [27]. To this end, the configuration spaces of each arm are augmented as if the whole multi-arm manipulator is viewed as a virtual single-arm manipulator whose configuration space has higher dimension. Since SAC is known to show superior exploration performance due to the entropy term in the objective function, this paper proposes how to apply SAC to designing of a path planning algorithm. In order to overcome the sparse reward problem, HER (Hindsight Experience Replay) [28] is employed to deal with generated episodes efficiently in training. Note that HER can convert the episodes without reward to episodes with reward by changing the target location.

The performance of the proposed SAC-based path planning is validated not only simulation but also experiment using two real open manipulators. The results show that the proposed method finds a shorter and smoother path for most scenarios due to enhanced exploration performance by SAC, and outperforms over the existing results such as PRM [29] and TD3 (Twin Delayed Deep Deterministic Policy Gradient)-based path planning [30].

2. Problem Setup and Preliminaries

2.1. Sampling-Based Path Planning for Robot Manipulator and Configuration Space

In sampling-based path planning for robot manipulators, the space of possible joint angles is referred to as configuration space $\mathbf{Q}$ (also called joint space) and is a subset of n-dimensional vector space $\mathbb{R}^n$ where n is the number of the joints of the robot manipulator. A point in the configuration space $\mathbf{Q}$ corresponds to values of joint angles of a robot manipulator [6,31,32]. The configuration space is comprised of two subsets: $\mathbf{Q}_{\text{collide}}$ and $\mathbf{Q}_{\text{free}}$. The robot manipulator does not collide with any obstacles or itself if its joint angles belong to $\mathbf{Q}_{\text{free}}$. If the values of the joint angles are in $\mathbf{Q}_{\text{collide}}$, it means that there is a collision between the robot manipulator and obstacles or itself.

For sampling-based path planning, random sampling for the continuous $\mathbf{Q}_{\text{free}}$ is done in order to obtain its discrete representation. Then, the discrete representation is modeled as a connected graph. From this construction, the nodes in the graph imply admissible configurations of the robot manipulators, and feasible paths between any two configurations are represented by connected edges in the graph.

For the purpose of describing the path planning problem, let $\bar{q}_t \in \mathbf{Q} \in \mathbb{R}^n$ denote the values of the joint angles of the manipulator. Also, let T represent the maximum number of the iteration in any path planning algorithm. This means that the algorithm either finds the shortest path successfully within T iterations for a given starting configuration $\bar{q}_{\text{init}} \in \mathbf{Q}_{\text{free}}$ and goal configuration $\bar{q}_{\text{goal}} \in \mathbf{Q}_{\text{free}}$, or decides that it is not possible to compute the shortest path for those configurations. With these definitions, when $\bar{q}_{\text{init}} \in \mathbf{Q}_{\text{free}}$ and $\bar{q}_{\text{goal}} \in \mathbf{Q}_{\text{free}}$ are given, path planning is to compute a valid continuous and shortest path on the graph linking $\bar{q}_{\text{init}}$ and $\bar{q}_{\text{goal}}$. Since the sampled nodes of $\mathbf{Q}_{\text{free}}$ are modeled as a connected graph, there always exists such a shortest path between any two nodes in $\mathbf{Q}_{\text{free}}$. Note that the path means the sequence of the state $\bar{q}_t$ such that $\bar{q}_0 = \bar{q}_{\text{init}}, \cdots, \bar{q}_{T_1} = \bar{q}_{\text{goal}}$ with $T_1 \leq T$, and both the sequence $\{\bar{q}_0, \cdots, \bar{q}_{T_1}\}$ and the line segment connecting any two consecutive state $\bar{q}_t$ and $\bar{q}_{t+1}$ belonging to $\{\bar{q}_0, \cdots, \bar{q}_{T_1}\}$ have to belong to $\mathbf{Q}_{\text{free}}$.

2.2. Collision Detection in Workspace Using the Oriented Bounding Box (OBB)

Even though the path planning algorithm mostly works in the configuration space $\mathbf{Q}$, the path generated by the path planning algorithm has to make sure that the collision does not occur when the robot moves in workspace $\mathbf{W}$. The workspace $\mathbf{W}$ is a space where an actual robot works, and is the subset of 3-dimensional Euclidean $\mathbb{R}^3$. It is important to have a collision detection method to confirm $\bar{q}_t \in \mathbf{Q}_{\text{free}}$ in the middle of running the path planning algorithm [6,33]. To detect the collision in the workspace $\mathbf{W}$, the oriented bounding boxes (OBB) [34] are employed for modeling all the 3D objects and finding the boundary in the 3D environment of the workspace including robots, obstacles, and ground. Simply speaking, any objects in the workspace are represented as boxes surrounding them in order to check collision.

Since objects in the workspace are modeled in the form of the OBB, the collision checking between two boxes has 15 cases: 3 faces of the first box, 3 faces of the second box, and 9 edge combinations between the first and second boxes [35]. By checking these 15 cases made by two OBBs, the collision detection between two boxes can be tested. Based on such a collision check for two boxes, the higher level collision checking between one robot and one obstacle can be implemented by repeated application of collision checking for two OBBs [36]. For the multi-arm manipulator case, each arm is considered as a moving obstacle to the others. Hence, again, collision checking in the workspace of the multi-arm manipulator can be tested using collision checking of OBBs for obstacles and robots.

2.3. Reinforcement Learning

Reinforcement learning is a decision making procedure for Markov decision process (MDP) [37] which is defined by the tuple $\{S, A, P, r, \gamma\}$ where S is the set of states, A the set of actions, P the transition probability, $r(s, a)$ the reward function, and γ the discount factor [38]. The transition probability $P(s'|s, a)$ is the probability that the current state $s \in S$ moves to the next state s' when the action $a \in A$ is applied to the environment at the current state $s \in S$. The policy of the agent denoted by $\pi(a|s)$ is the distribution of action a for each state $s \in S$. At each time step t, the agent chooses action $a_t \in A$ based on the policy $\pi : S \to A$. When the decided action is applied to the environment, the environment returns both the next state $s_{t+1} \in S$ and reward $r_{t+1} \in \mathbb{R}$ from the transition probability $P : S \times A \to \mathcal{P}(A)$ and the reward function $r : S \times A \to \mathbb{R}$. Conceptually, when the agent learns the optimal policy, the policy is determined such that it maximizes the expected return $E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}]$. This learning procedure is repeated until the expected return converges. However, since it is not possible to compute the expected return in practice, its estimated value, called optimal value function, is maximized. With this framework, there are two approaches: value-based and policy-based approaches. In the value-based approach, the optimal value function is estimated. For instance, deep Q-network (DQN) approximates the optimal value function using deep neural networks. When the optimal value function is estimated, the corresponding optimal policy is derived from the approximated value function. On the other hand, the policy-based approach (also called the policy gradient) computes the optimal policy directly from the agent's experience. Representatives of the policy gradient are REINFORCE, actor–critic method, deterministic policy gradient (DPG), deep DPG (DDPG), asynchronous advantage actor–critic (A3C), trust region policy optimization (TRPO), maximum a posteriori policy optimization (MPO) and distributed distributional DDPG (D4PG) [39–43]. In general, it is known that the policy gradient outperforms the value-based approaches, especially in the continuous action task [41,44].

In the reinforcement learning, the training performance depends heavily on sampled data consisting of the current state s_t, action a_t, next state s_{t+1} and reward r_{t+1}. To enhance sample efficiency, the agent saves the samples in the memory first and use the saved samples taken from the memory. Replay memory [45] and HER [28] are tailored methods for this. In this paper, a state-of-the-art policy gradient method, known as soft actor–critic (SAC) [27] with HER, is employed for the multi-arm manipulator path planning algorithm.

3. Method

3.1. Path Planning for the Multi-Arm Manipulator and Augmented Configuration Space

For the path planning of a multi-arm manipulator, it is assumed that there are m identical robot manipulators and each robot has n joints. Let $q_t^i \in \mathbb{R}^n$ denote the value of the joint angles of the ith manipulator at the tth iteration in the proposed deep reinforcement learning-based path planning algorithm, and $q_{t,j}^i \in \mathbb{R}$ represents the jth element of $q_t^i \in \mathbb{R}^n$, i.e., the value of the jth joint of the ith manipulator. Similarly to the path planning for a single robot manipulator, the path planning problem for the multi-arm manipulator is to find the shortest and collision-free path for a given starting configuration q_0 and goal configuration q_{goal} where

$$q_0 = \begin{bmatrix} q_{\text{init}}^1 \\ q_{\text{init}}^2 \\ \vdots \\ q_{\text{init}}^m \end{bmatrix}, \quad q_{\text{goal}} = \begin{bmatrix} q_{\text{goal}}^1 \\ q_{\text{goal}}^2 \\ \vdots \\ q_{\text{goal}}^m \end{bmatrix},$$

q^i_{init} and q^i_{goal} represent starting and goal configuration of the ith manipulator, respectively.

In order to design a path planning algorithm for the multi-arm manipulator, it is difficult to define $\mathbf{Q}_{\text{free}}$ and $\mathbf{Q}_{\text{collide}}$ for each arm if each arm is considered independently since the multi-arm collaborate (i.e., move together) in the same workspace and each arm is an obstacle to the others in the workspace. To cope with this situation, one remedy is to view the multi-arm manipulator as one virtual single-arm manipulator whose number of the joint is nm. Then, the virtual manipulator's movement is described by augmenting the configuration of each arm. In other words, the state of the virtual manipulator is defined by

$$q_t = \begin{bmatrix} q^1_t \\ q^2_t \\ \vdots \\ q^m_t \end{bmatrix} \in \mathbb{R}^{nm}, \;\; q^i_t = \begin{bmatrix} q^i_{t,1} \\ q^i_{t,2} \\ \vdots \\ q^i_{t,n} \end{bmatrix} \in \mathbb{R}^{n}, \;\; i \in \{1, 2, \cdots, m\}. \tag{1}$$

Consequently, the resulting augmented configuration space is defined by $\mathbf{Q}^a := \mathbf{Q}^a_{\text{free}} \bigcup \mathbf{Q}^a_{\text{collide}}$ where

$$\mathbf{Q}^a_{\text{free}} = \underbrace{\mathbf{Q}_{\text{free}} \times \cdots \times \mathbf{Q}_{\text{free}}}_{m \text{ times}} \subset \mathbb{R}^{nm} \;\text{ and }\; \mathbf{Q}^a_{\text{collide}} = \underbrace{\mathbf{Q}_{\text{collide}} \times \cdots \times \mathbf{Q}_{\text{collide}}}_{m \text{ times}} \subset \mathbb{R}^{nm},$$

$\mathbf{Q}^a_{\text{free}}$ denotes the collision-free space for q_t and $\mathbf{Q}^a_{\text{collide}}$ is the corresponding collision space. Then, the path planning problem for the multi-arm manipulator can be redefined by the path planning problem for the one virtual single-arm manipulator with a given starting configuration $q_0 \in \mathbf{Q}^a_{\text{free}}$ and goal configuration $q_{\text{goal}} \in \mathbf{Q}^a_{\text{free}}$.

3.2. Multi-Arm Manipulator Markov Decision Process (MAMMDP)

For the sake of applying deep reinforcement learning to path planning algorithm design for multi-arm manipulators, a multi-arm manipulator MDP (MAMMDP), i.e., the tuple $\{S, A, P, r, \gamma\}$, is defined. The structure of MAMMDP can be seen in Figure 1. The current state of m manipulators $q_t \in \mathbf{Q}^a_{\text{free}} \subset \mathbb{R}^{nm}$ is the n joint values of the m manipulators. The current action $a_t \in \mathcal{A}$ is defined as a change of the state in the augmented configuration space $\mathbf{Q}^a_{\text{free}}$. The action is generated by $a_t = f(e_t, q_t)$ where e_t follows the Gaussian distribution $\mathcal{N}(0, \sigma_t)$ with σ_t being the variance, and $f(e_t, q_t)$ generates a stochastic action using noise e_t and state q_t. Actually, function $f(\cdot)$ is a sampler. In other words, the action is sampled from a probability distribution. Then, the next state is calculated as the sum of the current state and action such as $\hat{q}_{t+1} = q_t + \alpha a_t + \epsilon_e$ where tuning parameter α is the geometric mean of the maximum variations of each joint at a time and $\epsilon_e \sim \mathcal{N}(0, \sigma_e)$ and ϵ_e is an environment noise. When $\hat{q}_{t+1} \in \mathbf{Q}^a_{\text{collide}}$ occurs, the next state becomes the previous state, i.e., it stays at the current state.

Such an iteration of computing, implementing the action, and obtaining the state and reward are repeated until the next state reaches the goal point q_{goal}, namely $q_{t+1} = q_{\text{goal}}$, or stops when the number of the iteration hits T where T is the predefined maximum number of the iteration. However, in practice, $q_{t+1} = q_{\text{goal}}$ rarely happens, at least, due to numerical problems. Hence, instead of that, a relaxed terminal condition $\|q_{t+1} - q_{\text{goal}}\| \le \eta \cdot \alpha$ is used where $\|\cdot\|$ denotes the norm of a vector and tuning parameter η meets $0 < \eta < 1$. Note that the terminal condition means that the iteration stops when the next state reaches a small neighborhood of the goal configuration.

When the next state is not the goal state, the corresponding reward is -1 while the reward is 0 if the agent reaches the goal. Hence, at the end of the iteration, if the agent does not reach the goal, then the total reward is $-T$. On the other hand, if the iteration ends at a certain iteration $T_1 < T$, then the total reward becomes $-(T_1 - 1)$. In summary, the state transition and reward function $r(q_t, a_t)$ are defined as follows.

$$q_{t+1} = \begin{cases} \hat{q}_{t+1}, & \text{if } \hat{q}_{t+1} \in \mathbf{Q}^a_{\text{free}} \\ q_t, & \text{if } \hat{q}_{t+1} \notin \mathbf{Q}^a_{\text{free}} \end{cases}, \qquad r_{t+1} = \begin{cases} 0, & \text{if } |q_{t+1} - q_{\text{goal}}| \le \eta \cdot \alpha \\ -1, & \text{if } q_{t+1} \in \mathbf{Q}^a_{\text{collide}} \\ -1, & \text{if } q_{t+1} \in \mathbf{Q}^a_{\text{free}} \end{cases} \tag{2}$$

The goal of reinforcement learning is to find the optimal policy maximizing the total reward that the agent gets in path planning iteration. If the iteration ends in a finite time $T_1 < T$, the agent can learn to find a path to reach the goal by maximizing the total reward since the graph is connected. Besides, by trying to reduce the ending T_1 during training, the agent can seek an as short path as possible. However, when the agent is not trained sufficiently, it is hard to find a goal since the action is determined mainly by only the exploration method (e.g., random action sampling). This means that there are many iterations which fail to reach the goal until the agent is well trained. In this case, there are only few state, action, and reward trajectories which can contribute to learning (i.e., higher total reward). MDP with such a problem is called MDP with sparse reward and the path planning problem for the robot arm manipulator exactly falls into MDP with sparse reward. To overcome this problem in this paper, a path planning algorithm is designed based primarily on both recently proposed soft actor–critic (SAC) [27] and hindsight experience replay (HER) [28]. SAC is famous for its outstanding exploration performance and HER enhances the sample efficiency in deep reinforcement learning for MDP with sparse reward. The details of the proposed algorithm are presented in the next subsection.

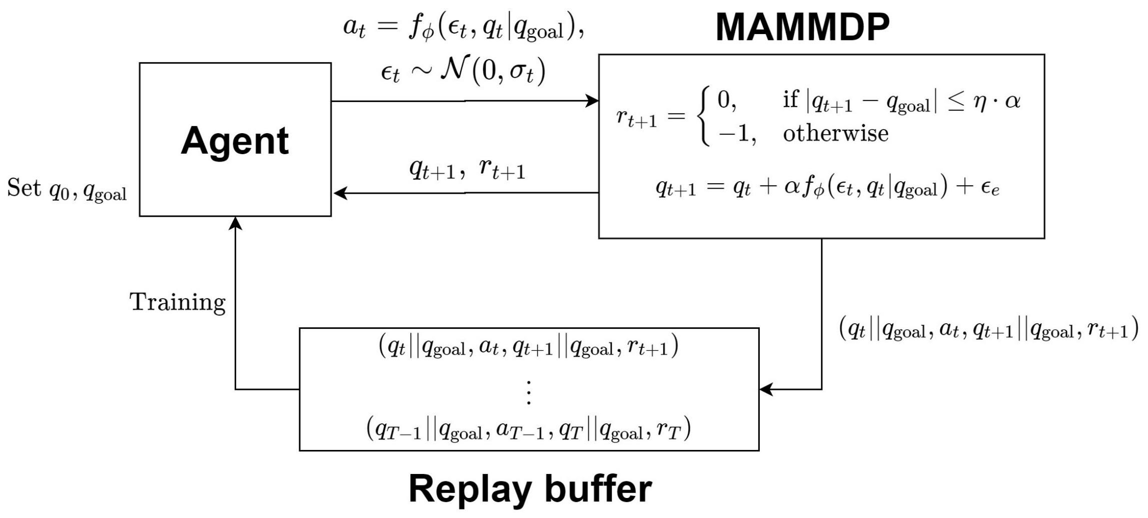

$a_t = f_\phi(\epsilon_t, q_t | q_{\text{goal}})$,
$\epsilon_t \sim \mathcal{N}(0, \sigma_t)$
MAMMDP
Agent
Set q_0, q_{goal}
$q_{t+1}, \ r_{t+1}$
$r_{t+1} = \begin{cases} 0, & \text{if } |q_{t+1} - q_{\text{goal}}| \le \eta \cdot \alpha \\ -1, & \text{otherwise} \end{cases}$
$q_{t+1} = q_t + \alpha f_\phi(\epsilon_t, q_t | q_{\text{goal}}) + \epsilon_e$
Training
$(q_t || q_{\text{goal}}, a_t, q_{t+1} || q_{\text{goal}}, r_{t+1})$
$(q_t || q_{\text{goal}}, a_t, q_{t+1} || q_{\text{goal}}, r_{t+1})$
$\vdots$
$(q_{T-1} || q_{\text{goal}}, a_{T-1}, q_T || q_{\text{goal}}, r_T)$
Replay buffer

Figure 1. Multi-Arm Manipulator Markov Decision Process (MAMMDP).

3.3. Soft Actor-Critic (SAC)

MAMMDP has a continuous action, i.e., the agent's action of MAMMDP is a continuous joint value and there are several policy gradient-based reinforcement learning methods for MDP with continuous action such as [27,41,46]. This section reviews briefly the main result of [27,47] and it is presented how to design a SAC-based path planning for MAMMDP. SAC is a maximum entropy reinforcement learning which maximizes not only the expected sum of rewards but also the entropy of the policy. Namely, SAC computes the optimal policy by maximizing the expected sum of entropy augmented reward defined by

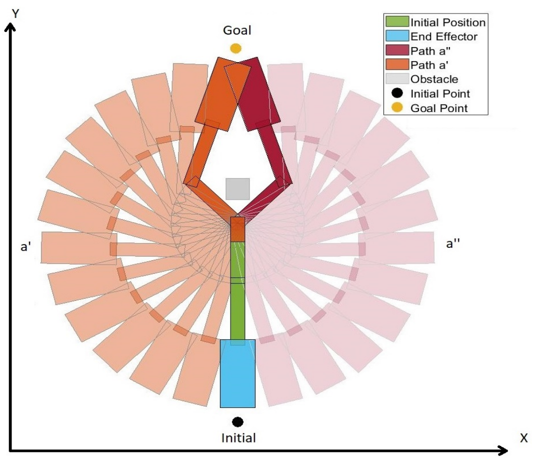
$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{q_t, a_t}[r(q_t, a_t) + \beta \mathcal{H}(\pi(a_t|q_t))], \quad \mathcal{H}(\pi(a_t|q_t)) = -\sum_{a_t} \pi(a_t|q_t) \log \pi(a_t|q_t), \tag{3}$$

where π is the policy which is updated to find the maximum total reward. β is the temperature parameter and it regulates the entropy term against the reward. $\mathcal{H}(\pi(a_t|q_t))$ denotes entropy.

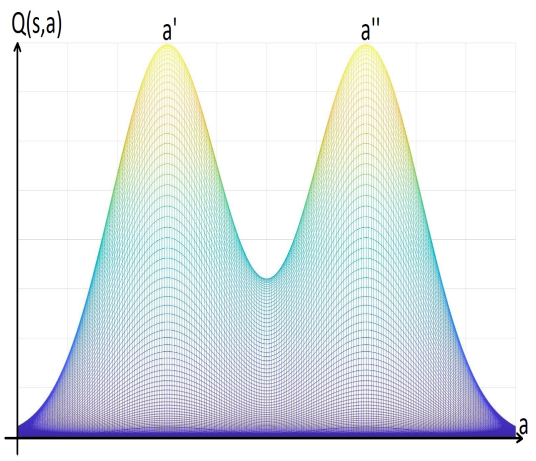
Usually, the distribution of the policy has small variance and its center is near a specific action that leads to the high return. On the other hand, the entropy in the objective of SAC makes the variance of the policy distribution increases. In the policy, the increased variance of the distribution means that the policy has more various actions that can be chosen. Accordingly, more explorations are carried out in this method. Because the multi-arm manipulator path planning is a multi-dimensional problem, enhancing the exploration plays a pivotal role in obtaining an efficient path planning algorithm. In SAC, the agent maximizes not only the expected return but also the entropy that leads to a better exploration. Because of this, the agent can acquire the optimal path of the multi-dimensional problem.

Remark 1. *As the policy is described by probability distribution in the case of a stochastic policy, the distribution of the policy should match that of Q-value. Besides, such a stochastic policy is heavily affected by the exploration. Because SAC is a maximum entropy reinforcement learning framework, it not only enhances the exploration but also finds multiple optimal policies, which is possible due to the fact that the distribution functions of the Q-value and the policy are getting match as the training is going on by SAC. In the path planning problem, in general, there can be multiple solutions for the optimal path.*

For example, in Figure 2a, there are the initial point, goal point, and an obstacle in between them. In such a setting, it is obvious that there are two shortest paths between q_{init} and q_{goal}. It is difficult for a deterministic policy to capture these two solutions. In contrast, the problem with multiple optimal solutions can be tackled by a stochastic policy with entropy augmented reward. Then, the probability distribution of the policy represents the information of two optimal solutions like in Figure 2b. Because of this, the stochastic policy such as SAC can find all optimal paths of a problem with multiple optimal solutions. This is why the SAC-based path planning can be effective for the path planning for a multi-arm manipulator which is highly dimensional.



(**a**) Multiple solutions in the path planning problem



(**b**) Multi-modes in action distribution

Figure 2. Multiple solutions in the path planning problem.

With the augmented reward function, the soft value function and soft Q-value function (or soft action value function) are defined respectively as follows:

$$V(q_t) = \mathbb{E}_{a_t \sim \pi}[Q(q_t, a_t) - \beta \log \pi(a_t|q_t)], \tag{4}$$

$$Q(q_t, a_t) = r(q_t, a_t) + \gamma \mathbb{E}_{q_{t+1} \sim p}[V(q_{t+1})]. \tag{5}$$

These functions are evaluated at q_t and a_t, and tell about how much reward can be obtained in the future. Soft Q-value $Q(q_t, a_t)$ is used to train the agent's policy and the soft state value $V(q_t)$ is needed to estimate the soft Q-value. In SAC, the policy, the soft state value, and the soft Q-value functions are approximated by deep neural networks (DNNs). Each of them is parameterized by ϕ, ψ, and θ, respectively. Besides, there exists another target network with parameter $\bar{\psi}$. The target network is used to improve learning performance for the soft Q-value and makes the learning more stable. In addition, the technique using the double Q-value functions is employed [27,46,48]. Consequently, there are five DNNs in SAC: the parameterized policy $\pi_\phi(a_t|q_t)$, the soft state value $V_\psi(q_t)$ with target $V_{\bar{\psi}}(q_t)$ and the two soft Q-values $Q_{\theta_{1,2}}(q_t, a_t)$. Finally, the action is sampled from $f_\phi(e_t, s_t)$, i.e., $a_t = f_\phi(e_t, s_t)$.

To find the optimal policy, soft-Q value and soft state value, the stochastic gradient descent method is applied to their objective functions. The DNN for the soft state value is trained to minimize the mean squared error with estimated soft state value given by

$$J_V(\psi) = \mathbb{E}_{q_t}[\frac{1}{2}(V_\psi(q_t) - \mathbb{E}_{a_t}[\min_{k=1,2} Q_{\theta_k}(q_t, a_t) - \beta \log \pi_\phi(a_t|q_t)])^2]. \tag{6}$$

In this function, the minimum of the soft Q-value is taken over two Q-value functions parameterized by θ_1 and θ_2. This helps to avoid overestimation that gives high ratings to inappropriate Q-values [46]. The soft Q-value is trained to predict how much return can be obtained from pair (q_t, a_t) and this is also done by minimizing the Bellman equation given by

$$J_Q(\theta_{k=1,2}) = \mathbb{E}_{q_t,a_t}[\frac{1}{2}(Q_{\theta_{k=1,2}}(q_t, a_t) - (r(q_t, a_t) + V_{\bar{\psi}}(q_{t+1})))^2]. \tag{7}$$

Both θ_1 and θ_2 are trained with their Q-values. The minimum of these Q-values is also used for the policy network learning. The objective function of the policy network is the information projection between the distribution of the current policy and the distribution of the minimum Q-value as follows.

$$J_\pi(\phi) = \mathbb{E}_{q_t,a_t}[\log \pi_\phi(a_t|q_t) - \min_{k=1,2} Q_{\theta_k}(q_t, a_t)]. \tag{8}$$

As a matter of fact, this objective function is a simplified Kullback-Leibler (KL) divergence between the policy and Q-value [27]. After minimizing this function with respect to parameter ϕ, the distribution of the policy becomes proportional to the distribution of Q-value. At each training step, after training the other parameters, the parameter of the target soft state value $\bar{\psi}$ is updated according to $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$ by a tuning parameter $\tau \in [0, 1]$. Since SAC is based on off-policy actor–critic [27,44], we do not need to use the current policy for training the networks. Instead, the transition tuples $(q_t, a_t, q_{t+1}, r(q_t, a_t))$ are collected in every iteration and then, stored in experience replay memory $\mathcal{D}$ [45]. At the beginning of the training procedure, a bundle of these tuples is sampled and they are used to compute the expectation in the objective functions.

3.4. Hindsight Experience Replay (HER)

In the MAMMDP under consideration, the agent does not receive negative rewards only when the goal is reached. As mentioned before, since the MAMMDP is an MDP with sparse reward, the agent suffers from low sample efficiency. To improve the sample efficiency, the hindsight experience replay (HER) [28] method is used to deal with the measured samples. Suppose that the iteration in the algorithm ends without reaching the goal state, and let the corresponding trajectory becomes a failed episode $e = [(q_0, a_0), (q_1, r_1, a_1, \cdots, (q_T, r_T))]$. Since the agent does not arrive at the goal state in this episode, it follows that $q_T \neq q_{\text{goal}}$, which means that the agent does not learn much from this episode. To resolve this problem, HER defines a new goal state q'_{goal}, chooses one state in the failed episode, let say $q_{t'}$, $t' \in \{1, 2, \cdots, T\}$, and sets $q'_{\text{goal}} = q_{t'}$. Then, the modified episode $e' = [(q_0, a_0), (q_1, r_1, a_1, \cdots, (q_{t'}, r'_{t'}))]$ becomes a successful episode since $q_{t'} = q'_{\text{goal}}$ is achieved where $r'_{t'} \equiv 0$. Then, this modified episode can improve the learning performance and this is how HER enhances the sample efficiency.

The proposed SAC-based path planning algorithm is depicted in Figures 3 and 4. Also, the details of the implementation of the proposed method can be found in Appendix A.

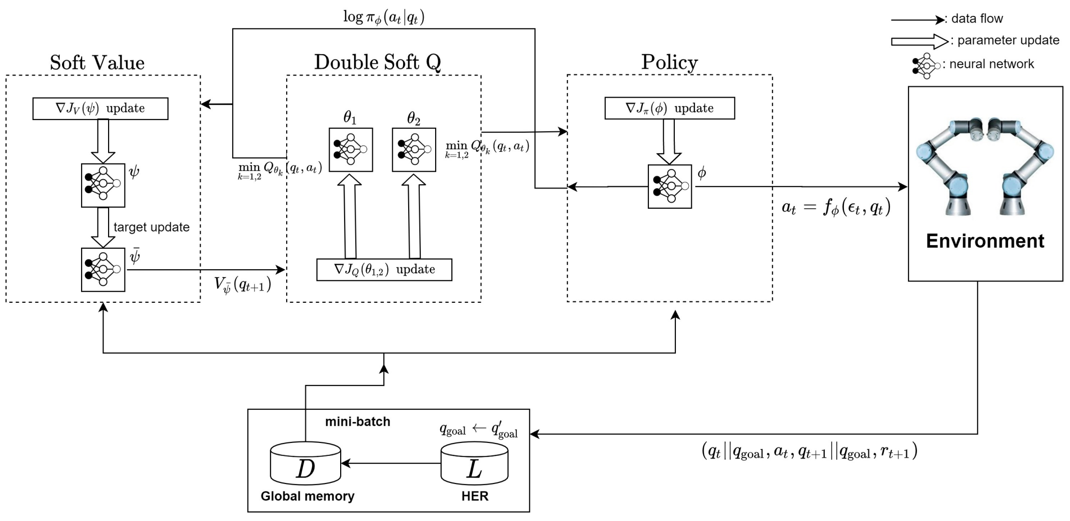

$\log \pi_\phi(a_t|q_t)$
: data flow
: parameter update
: neural network
Soft Value
Double Soft Q
Policy
$\nabla J_V(\psi)$ update
ψ
target update
$\bar{\psi}$
θ_1
θ_2
$\min_{k=1,2} Q_{\theta_k}(q_t, a_t)$
$\min_{k=1,2} Q_{\theta_k}(q_t, a_t)$
$\nabla J_Q(\theta_{1,2})$ update
$V_{\bar{\psi}}(q_{t+1})$
$\nabla J_\pi(\phi)$ update
ϕ
$a_t = f_\phi(\epsilon_t, q_t)$
Environment
mini-batch
$q_{\text{goal}} \leftarrow q'_{\text{goal}}$
D
L
Global memory
HER
$(q_t||q_{\text{goal}}, a_t, q_{t+1}||q_{\text{goal}}, r_{t+1})$

Figure 3. Architecture of the proposed SAC-based path planning algorithm.

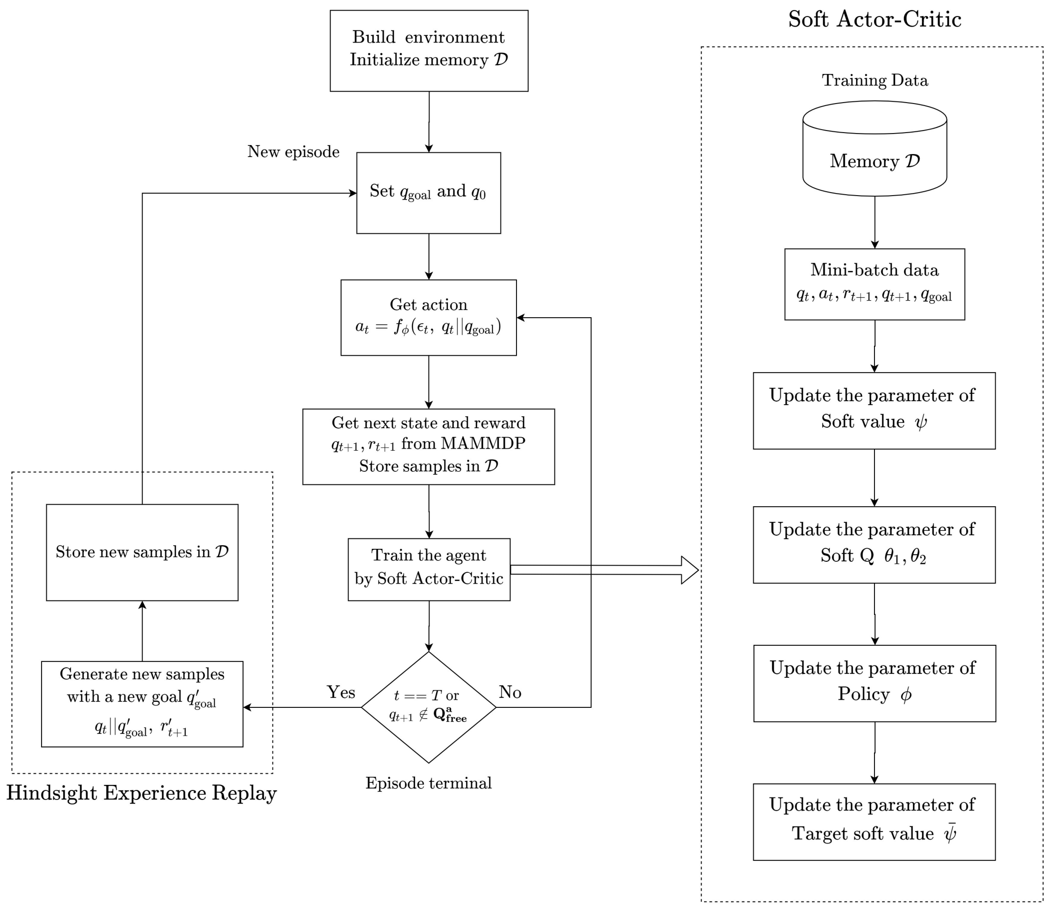

Soft Actor-Critic
Build environment
Initialize memory $\mathcal{D}$
New episode
Set q_{goal} and q_0
Get action
$a_t = f_\phi(\epsilon_t, \; q_t||q_{\text{goal}})$
Get next state and reward
q_{t+1}, r_{t+1} from MAMMDP
Store samples in $\mathcal{D}$
Train the agent
by Soft Actor-Critic
$t == T$ or
$q_{t+1} \notin \mathbf{Q}^{\mathbf{a}}_{\mathbf{free}}$
Yes
No
Episode terminal
Store new samples in $\mathcal{D}$
Generate new samples
with a new goal q'_{goal}
$q_t||q'_{\text{goal}}, \; r'_{t+1}$
Hindsight Experience Replay
Training Data
Memory $\mathcal{D}$
Mini-batch data
$q_t, a_t, r_{t+1}, q_{t+1}, q_{\text{goal}}$
Update the parameter of
Soft value ψ
Update the parameter of
Soft Q θ_1, θ_2
Update the parameter of
Policy ϕ
Update the parameter of
Target soft value $\bar{\psi}$

Figure 4. Flow chart of the proposed SAC-based path planning algorithm.

4. Results and Discussion

In this section, the proposed SAC-based path planning algorithm is applied to two real 3-DOF open manipulators. The parameters of the 3-DOF manipulator are given in Table 1. To see the details of the 3-DOF open manipulator, see http://en.robotis.com/model/page.php?co_id=prd_openmanipulator.

Table 1. Parameters of the 3-DOF manipulator.

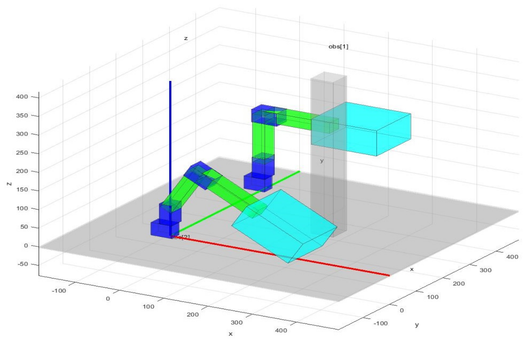
Name	Value	Notation
The number of joints for each manipulator	3	n
The number of manipulators	2	m
Dimension of $\mathbf{Q}^a_{\text{free}}$	6	$n \cdot m$
Joint maximum	(140, −45, 150, 140, −45, 150)	
Joint minimum	(−140, −180, 45, −140, −180, 45)	

The parameters used in the SAC-based path planning are described in Table 2.

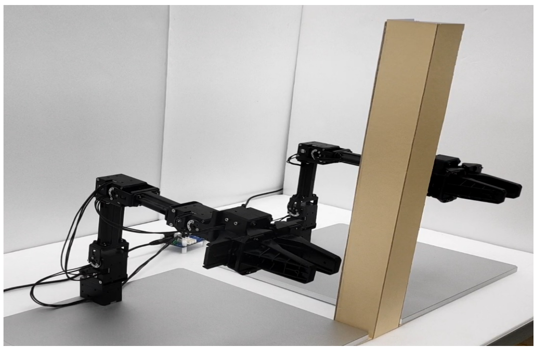
Table 2. Tuning parameters for the designed SAC with HER.

Name	Value	Notation
Policy network size	12∗800∗500∗400∗400∗300∗6	ϕ
Soft Q network size	18∗800∗500∗400∗400∗300∗1	$\theta_{1,2}$
Soft value network size	12∗800∗500∗400∗400∗300∗1	ψ
Learning rate	0.0001	
Replay memory size	10^6	$\mathcal{D}$
Episode maximum step	100	T
Soft value target copy rate	0.005	τ
Mini batch size	512	m
Environment noise deviation	0.002	ϵ_e
Action step size	0.3813	α
Goal boundary	0.2	η
Dicount factor	0.98	γ
Entropy temperature parameter	0.2	β

For the experiment, the workspace in Figure 5 is considered. The size of the workspace is 90 cm × 60 cm × 47 cm. The bar in between two manipulators is an obstacle. Figure 5a represents the visualization of the workspace in Matlab and Figure 5b shows the actual workspace. The problem to be solved is to devise a path planning algorithm using the proposed method for each arm in this setting such that the shortest paths for each arm are computed when arbitrary start and goal positions for the end-effect of each arm are given. Note that the path planning for this problem setup is nontrivial since there is a common workspace in workspaces of each arm in addition to the obstacle.



(**a**) Workspace in matlab



(**b**) Actual workspace

Figure 5. The workspace of robots.

In the training process, 300,000 episodes are implemented to train the networks in the proposed SAC-based path planning algorithm. In each episode, arbitrary start and goal locations belonging to the workspace are given and the agent is trained to find the shortest path between them by the proposed algorithm. Figure 6 shows that success ratio of the 300,000 episodes.

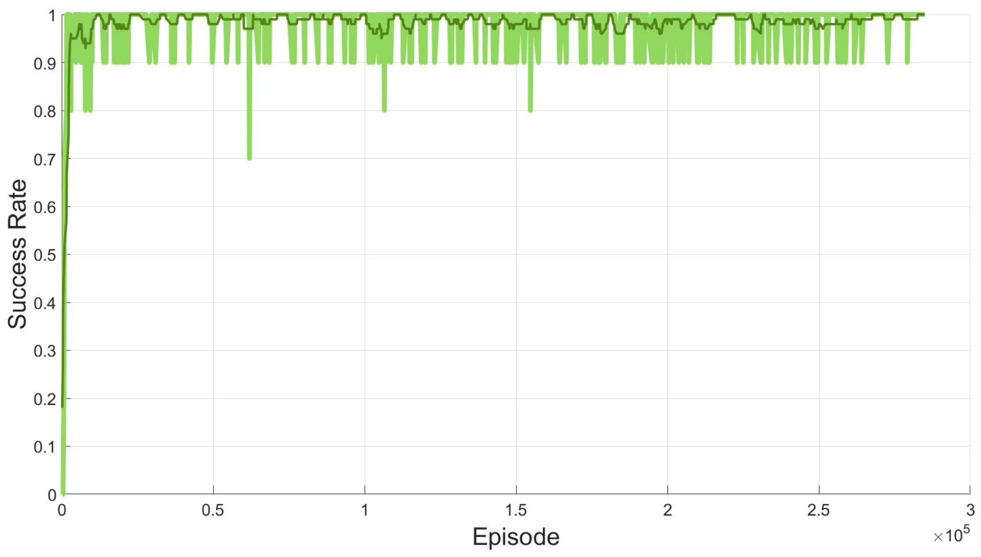


Figure 6. Success ratio by the proposed SAC-based path planning for two open manipulators.

The success means that the path planning algorithm finds the shortest path successfully. In Figure 6, the green line denotes the moving average of success ratio of every 10 episodes where the success ratio is set to 1 when the shortest path is computed successfully, and 0 otherwise. The thick line is the moving average of the green lines. As seen in Figure 6, the proposed path planning algorithm finds the path without fail in most cases.

In addition to the success ratio, the reward is also an important aspect to evaluate the training performance of any reinforcement learning-based algorithm. In the proposed algorithm, the reward value can be interpreted as the optimal time generating the path due to the definition of the reward Function (2). Figure 7 depicts the reward for the 300,000 episodes.

In Figure 7, the light blue line denotes the actual reward and the thick blue line describes the moving average of the light blue line. In view of Figure 7, it can be seen that the reward converges very quickly as the episode increases, which implies the algorithm finds the shortest path efficiently and consistently. Figures 6 and 7 demonstrate that the agent is trained well by the proposed SAC-based path planning algorithm such that it not only reaches the given goal mostly but also maximizes the return due to the quickly converged reward, which means that the training process is done successfully.

To see the advantage of the proposed method more, we implement other algorithms in the training process and plot the data in one graph for comparison. Figure 8 shows the training results made by three different path planning algorithms based on deep reinforcement learning. The light blue line is the actual reward from TD3 (Twin Delayed Deep Deterministic Policy Gradient) and the thick blue line is the moving average of the light blue line. Then, the orange line indicates SAC without entropy ($\beta = 0$) and the purple line is the original SAC with entropy. For SAC without entropy, the moving average is in the thick orange line and the moving average of SAC is in the thick purple line. Basically, because β is equal to zero in SAC without entropy, the objective function of SAC without entropy is exactly the same as the objective function of TD3. At the beginning of the training, we can see that SAC has a late transient response than TD3 and SAC without entropy. This is because SAC has an entropy factor in the objective function. Moreover, the TD3 and SAC without entropy have the same objective function that only maximizes the expected reward without the entropy. However, at the steady-state, SAC can reach a higher reward value than the other algorithms. This means that the trained actor from SAC can find shorter path than the other algorithms. This is because SAC aims to maximize not only the expected reward but also the entropy.

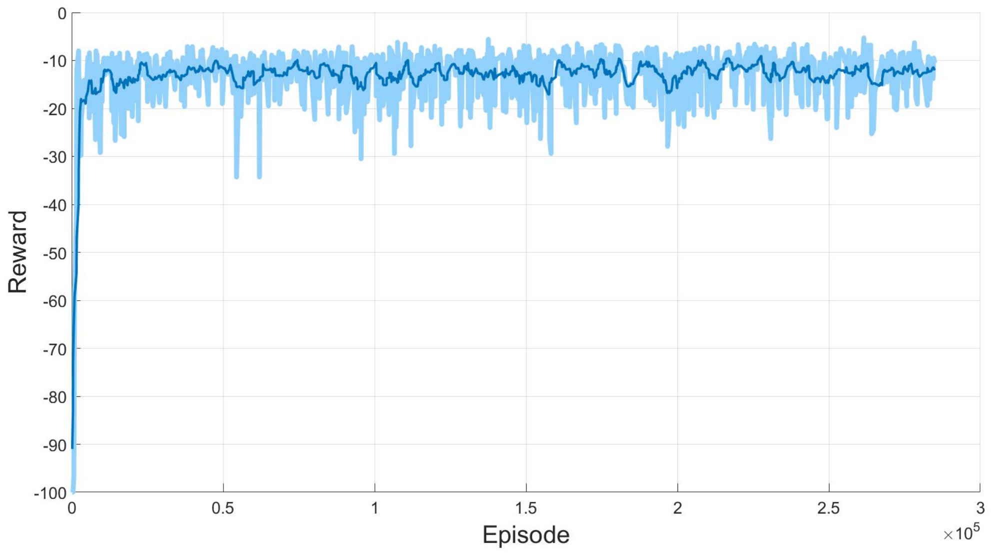


Figure 7. Reward from learning for two 3-DOF manipulators.

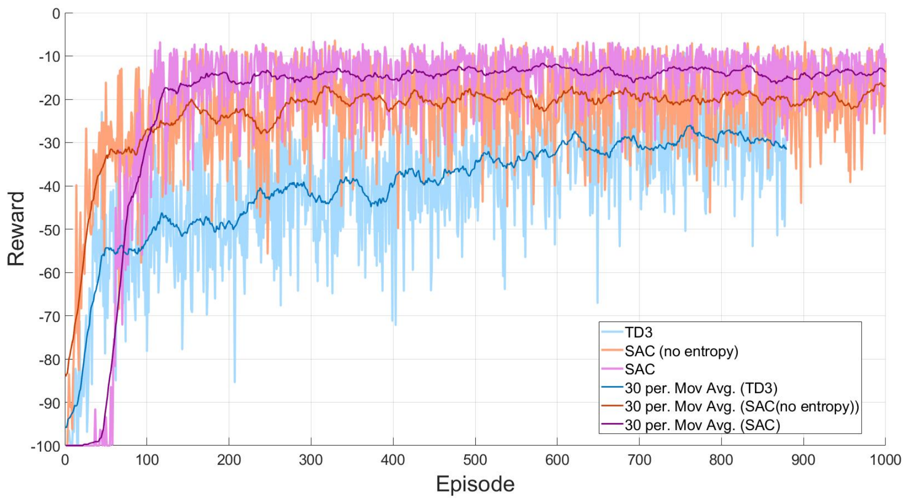


Figure 8. Comparison of the training performances between TD3 and SAC.

Once the actor is trained well, the actor-network can be used to compute the shortest path for a given arbitrary start and goal location in the workspace. As the output of the trained actor is only the mean of the stochastic policy, the output does not have any noise from the sampling distribution. To be specific, in testing step, $(q_{\text{init}}, q_{\text{goal}})$ is injected into the trained actor in the beginning. Then, its output is the mean action (without noise) which is applied to the environment. The measurement, including the environment noise, is the next state which is injected into the actor-network together with q_{goal}. Then, the actor-network generates the action again. This procedure is repeated until the goal state is reached. In other words, the trained actor generates the following path to the goal state.

$$\underbrace{(q_{\text{init}}, q_{\text{goal}})}_{\text{Input to the actor}} \rightarrow \underbrace{a_0}_{\text{Output of the actor}} \rightarrow \underbrace{q_1}_{\text{Measurement}} \rightarrow \underbrace{(q_1, q_{\text{goal}})}_{\text{Input to the actor}} \rightarrow a_1 \rightarrow q_2 \cdots (q_t, q_{\text{goal}}) \rightarrow \cdots \rightarrow (q_{\text{goal}}, q_{\text{goal}}).$$

Figure 9 depicts the path inference procedure.

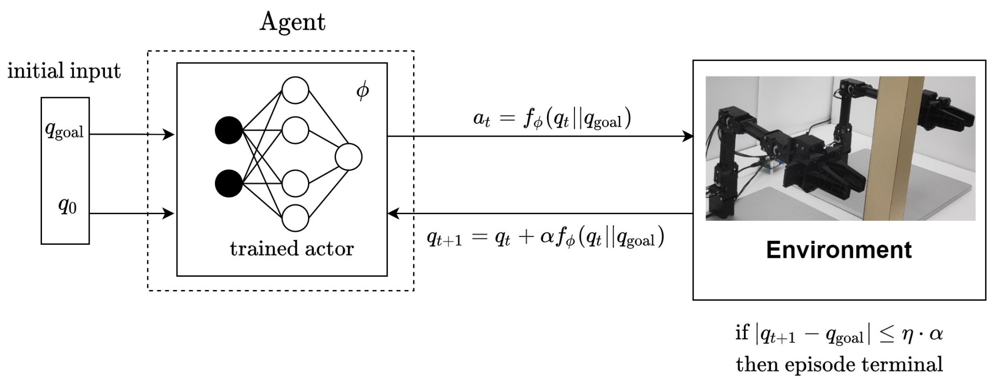

Agent
initial input
q_{goal}
q_0
ϕ
trained actor
$a_t = f_\phi(q_t||q_{\text{goal}})$
$q_{t+1} = q_t + \alpha f_\phi(q_t||q_{\text{goal}})$
Environment
if $|q_{t+1} - q_{\text{goal}}| \leq \eta \cdot \alpha$
then episode terminal

Figure 9. Path generation using the trained actor DNN.

Figure 10 shows an example of the start position q_{init} and goal position q_{goal} in the workspace where the green robot position is the initial position and the light yellow denotes the goal position. As explained before, if q_{init} and q_{goal} are given to the trained agent, the shortest path is computed.

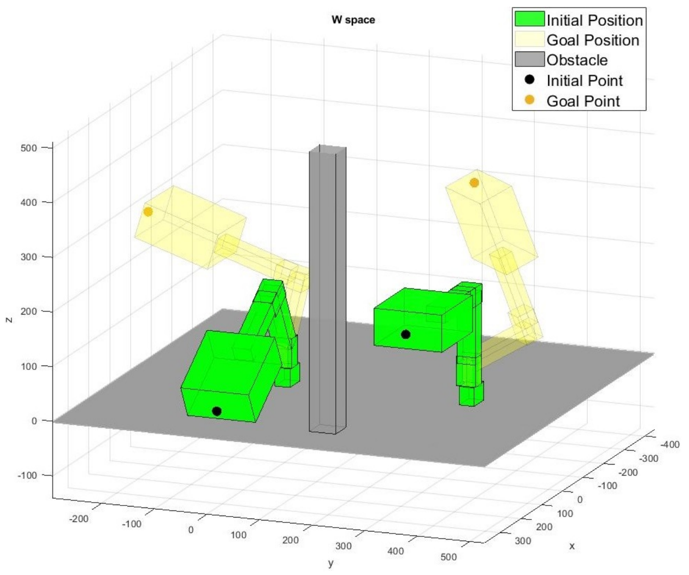
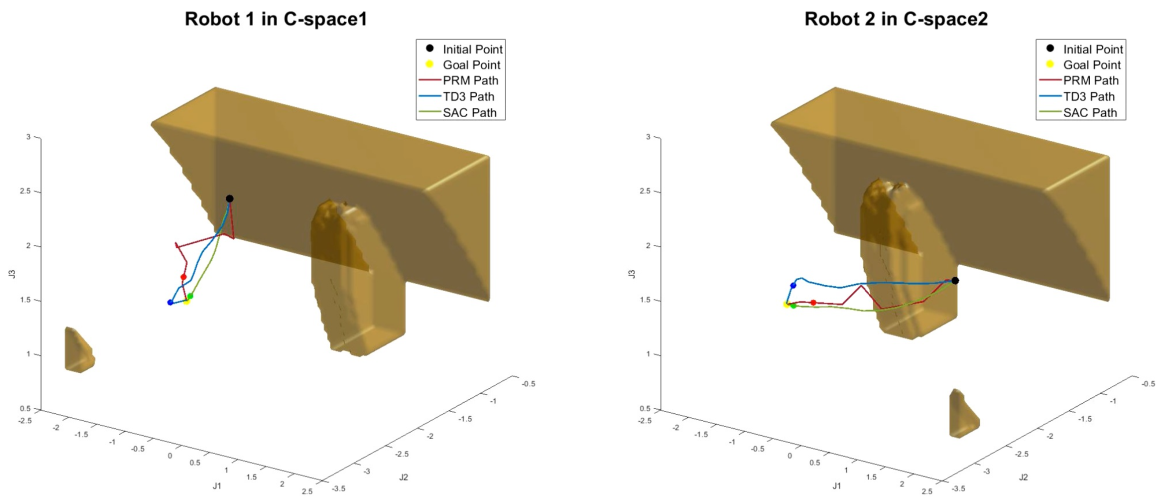


Figure 10. Initial and goal positions in the workspace.

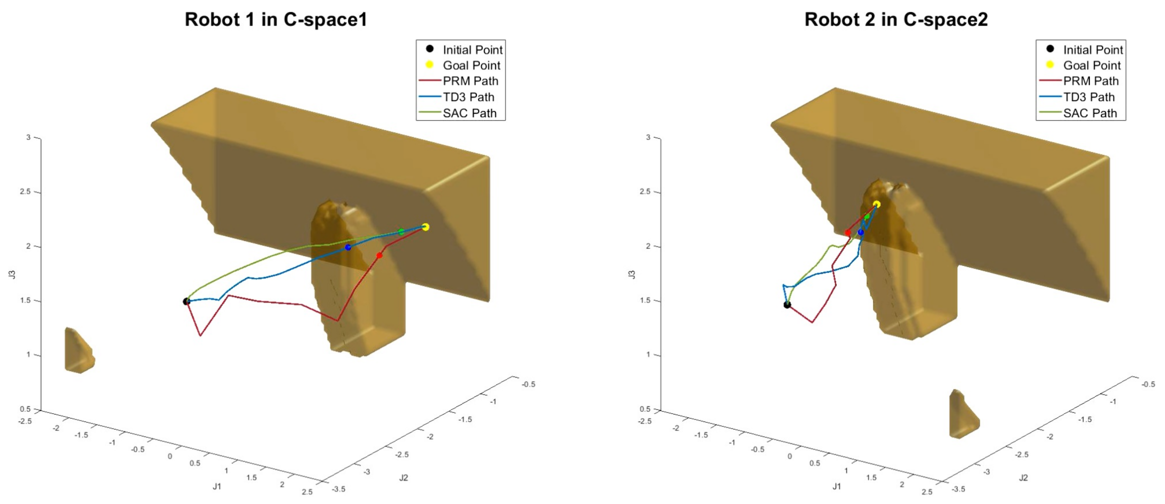
Figure 11 depicts the configuration space ($\subset \mathbb{R}^3$) of each arms extracted from the augmented configuration space ($\subset \mathbb{R}^6$). The left figure in Figure 11 shows the configuration space of the first arm and the right figure that of the second arm. Figure 11 shows several examples of resulting paths computed by the actor for given arbitrary start and goal locations in the configuration space. The two configuration spaces of each robot are used for visualizing the generated path. The lines in Figure 11 represent the joint movement of robots. In the configuration space, there are two kinds of collision space: the static collision space as a static obstacle and dynamic collision space as a moving obstacle (the other robot). In Figure 11, the static obstacle is represented by the brown area of the configuration space. The video clips for the simulation and experiment can be found at https://sites.google.com/site/cdslweb/publication/sacpath.

For comparison, the other method like PRM (35,000 sampled point graph) and TD3 with HER also generate a path with the same task. In Figure 11, the red lines denote the result by PRM method and the blue lines by TD3 with HER, and the green lines by the proposed method. The yellow points describe the goal position of each robot and the black points tell the start position of each robot. As shown in

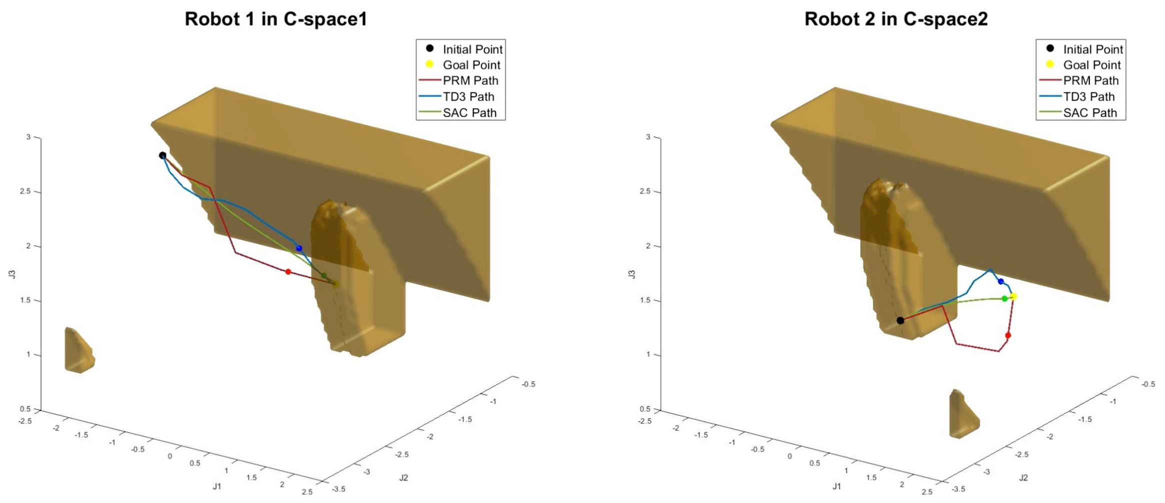
Figure 11, the proposed path planning using SAC with HER method leads to smoother and shorter paths than the other algorithms.



(**a**) Scenario 1



(**b**) Scenario 2



(**c**) Scenario 3

Figure 11. Path generation by SAC with HER for arbitrary initial and goal points in each C-space robot.

To see the quality of the proposed method, 100 simulations of the arbitrary initial and goal points are implemented using the PRM, TD3 with HER, and SAC with HER simultaneously. Since the result paths can be interpreted as performance, the 100 generated paths are used for comparison in Figure 12. As seen in Figure 12, the proposed path planning using SAC with HER always finds the shortest path for all simulation cases. For the result shown in Figure 12, it takes 0.1385 s on average computing the shortest path for a start point and goal point. This means that the proposed SAC-based path planning can generate the optimal path quickly so that it works for the multi-arm manipulator in real-time such as industrial application. Note that the training is done off-line.

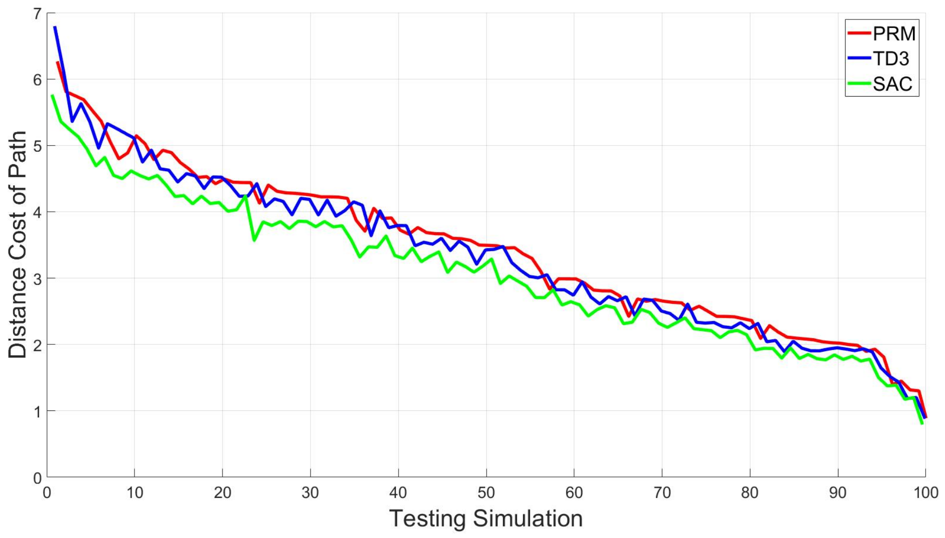


Figure 12. Comparison of paths by PRM, TD3 with HER and the proposed method.

For better comparison, the paths are calculated for 100 simulations using different algorithms and are drawn in Figure 13.

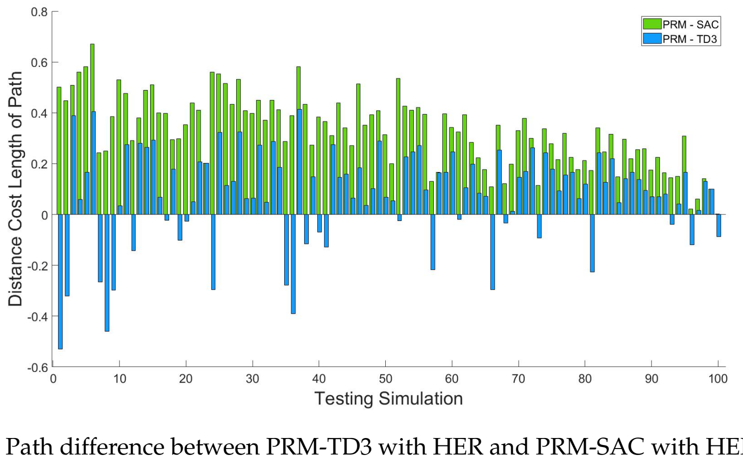


Figure 13. Path difference between PRM-TD3 with HER and PRM-SAC with HER by PRM.

Table 3. Comparison of the proposed result with existing methods.

Method	Average Path Cost	Cost Percentage	Roughness
PRM	3.4162	100%	0.4153
TD3	3.3402	97.8%	0.0437
SAC	3.0862	90.3%	0.0206

From Figure 13, we can see that SAC- or TD3-based path planning is better than PRM, and SAC-based algorithm is better than TD3-based algorithm. In Table 3, three path planning methods are compared. The average path cost is the average of the path length from testing simulation in Figure 12, and the cost percentage is another ratio of the path cost where PRM's average path cost is defined as 100%. As shown in Table 3, SAC leads to the lowest cost compared with others. The roughness is a metric representing how smooth the path is [49]. It is evaluated by the mean squared error with second derivative of the path. In other words, the roughness is defined by $l_{\text{roughness}}(q_1, q_2, \cdots, q_T) = \frac{1}{T}\sum_{t=1}^{T}(\frac{d^2q_t}{dt^2})^2$. As shown in Figure 11 and Table 3, the generated path by SAC is smoothest among the others.

Such good performance of the proposed SAC-based algorithm is due to its better exploration performance coming from maximizing the entropy. In the maximum entropy framework, the agent explores the environment to maximize the return and the entropy. Because of this reason, the algorithm can find the optimal path in any task. However, in deterministic policy like TD3, the agent only maximizes the expected reward, which can lead to poor exploration performance.

In view of all simulation and experimental results, it is confirmed that the proposed method generates a shorter and smoother path than other algorithms. To be specific, notice that the optimal path of the proposed method is shorter by 9.66% than that of PRM on average. Compared with the TD3 with HER, on average, the paths by the proposed method are 7.6% shorter than those by TD3 with HER.

5. Conclusions and Future Work

This paper presents a deep reinforcement learning-based path planning algorithm for the multi-arm manipulator. In order to solve the high-dimensional path-planning problem, SAC (Soft Actor-Critic)-based algorithm is proposed. To deal with the multi-arm efficiently in configuration space, configuration spaces of each arm are augmented, and HER (Hindsight Experience Replay) is employed to enhance sample efficiency. Both the simulation and experiment show that the proposed algorithm finds the shortest path for arbitrary start and goal positions and the generated paths are shorter and smoother than those generated by existing results. Such results are made due to better exploration performance yielded by the entropy term in the objective function of SAC. Since the agent is trained off-line, the trained agent can generate the shortest path for an arbitrary start and goal positions quickly which means that the proposed algorithm can be used in real-time application.

This paper assumed that the work environment is static. Namely, the obstacles in the workspace do not change during the robot arm manipulator operation. Hence, a natural future work is the path planning for dynamic environment. For instance, if obstacles are moving in workspace, it is nontrivial to define Markov decision process in order to design a path planning algorithm using reinforcement learning.

Author Contributions: E.P. and M.K. 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., J.-H.P., and J.-H.B. 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)(20005024, Development of intelligent robot technology for object recognition and dexterous manipulation in the real environment) funded by the Ministry of Trade, Industry & Energy(MOTIE, Korea), and by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2019R1A6A1A03032119).

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

Abbreviations

The following abbreviations are used in this manuscript:

MAMMDP	Multi-Arm Manipulator Markov Decision Process
SAC	Soft Actor-Critic
HER	Hindsight Experience Replay
AI	Artificial Intelligence
FMMs	Fast Marching Methods
PRM	Probabilistic Road Map
RRT	Rapid exploring Random Trees
DNN	Deep Neural Network
TD3	Twin Delayed Deep Deterministic Policy Gradient
MDP	Markov Decision Process
DOF	Degree of Freedom
OBB	Oriented Bounding Boxes
DQN	Deep Q-Network
DPG	Deterministic Policy Gradient
DDPG	Deep Deterministic Policy Gradient
A3C	Asynchronous Advantage Actor-Critic
TRPO	Trust Region Policy Optimization
MPO	Maximum a Posteriori Policy Optimisation
D4PG	Distributed Distributional Deep Deterministic Policy Gradient
KL	Kullback-Leibler

Appendix A

In this section, the proposed algorithm is explained in terms of implementation. The proposed reinforcement learning-based path planning algorithm starts from goal-conditioned reinforcement learning. In this setting, the policy of the agent is defined like $\pi(a_t|\ q_t||q_{\text{goal}})$ which is probability of the action a_t where current state q_t and goal state q_{goal} are given. $q_t||q_{\text{goal}}$ is the concatenated vector of the joint values and is the input of the policy $\pi(\cdot,\cdot)$ where $||$ denotes the concatenation. This setting is the same as that of the original HER (Hindsight Experience Replay) algorithm [28]. Also, the soft Q-value function and the soft state-value function depend not only on a state-action pair but also on a goal where the soft Q-value function is $Q(q_t||q_{\text{goal}}, a_t)$ and the soft state-value function is $V(q_t||q_{\text{goal}})$.

In the HER, the additional goal $q'_{\text{goal}} \in \{q_1, q_2, \cdots, q_T\}$ is selected to generate a new sample $q_t||q'_{\text{goal}}$ at the end of every episode. The additional goal is randomly chosen from $\{q_1, q_2, \cdots, q_T\} \subset \mathbf{Q}^a_{\text{free}}$ and $q'_{\text{goal}} \notin \mathbf{Q}^a_{\text{collide}}$. When the agent is trained with this new sample, the sample efficiency of training process can be improved. Since the reward function of MAMMDP is based on a sparse reward setting and has no information on the direction or distance to the goal state, it is extremely difficult to train the agent without HER algorithm. This gets worse as the dimension of the path planning problem increases. By using this technique, we can increase the number of additional goals at each episode to speed up the training. However, when we compare the performance between TD3 and SAC in Figure 8, only one additional goal is used in every episode.

By utilizing reparameterized sampler $f_\phi(\epsilon_t, q_t||q_{\text{goal}})$ and without using original stochastic policy $\pi_\phi(a_t|\ q_t||q_{\text{goal}})$, SAC produces a continuous action as its output [27]. Then, for the sampling noise, ϵ_t is given from the Gaussian distribution where $\epsilon_t \sim \mathcal{N}(0, \sigma_t)$ and σ_t is a variance. To be specific, the outputs of the policy network are mean μ_ϕ and variance σ_t. According to the objective Functions (6) and (8), we need to calculate $\mathbb{E}_{a_t} \log \pi_\phi(a_t|\ q_t||q_{\text{goal}})$ where $a_t = f_\phi(\epsilon_t, q_t||q_{\text{goal}})$. This is nothing but log-likelihood with the probability of action $\pi_\phi(a_t|\ q_t||q_{\text{goal}})$. Since a_t is sampled from Gaussian distribution, this log-likelihood is given by

$$\mathbb{E}_{a_t} \log \pi_\phi(a_t|\ q_t||q_{\text{goal}}) = -\frac{l}{2}\log(2\pi) - \frac{l}{2}\log(\sigma_t^2) - \frac{1}{2\sigma_t^2}\sum_{i=1}^{l}(a_i - \mu_\phi(q_t||q_{\text{goal}})), \tag{A1}$$

where l is the number of samples. Variance σ_t can be viewed as a design parameter or can be tunable via learning. In this paper, it is fixed as a constant. Figure A1 describes how the action is sampled.

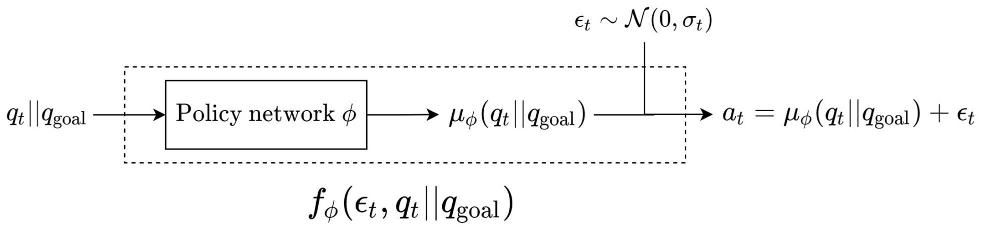

$\epsilon_t \sim \mathcal{N}(0, \sigma_t)$
$q_t||q_{\text{goal}}$
Policy network ϕ
$\mu_\phi(q_t||q_{\text{goal}})$
$a_t = \mu_\phi(q_t||q_{\text{goal}}) + \epsilon_t$
$f_\phi(\epsilon_t, q_t||q_{\text{goal}})$

Figure A1. Reparameterized sampler for the continuous action sampling.

Soft Actor-Critic, unlike the original policy gradient algorithm, maximizes the expected sum of entropy augmented reward in the Equation (3). The entropy in the reward function can be evaluated by the Equation (A1) and this entropy augmented reward Function can be seen in the objective Functions (6)–(8). The order of optimizing objective Functions (6) and (7) is not important, but the optimization of objective Function (8) must be done after optimization of objective Functions (6) and (7). That is because the policy has to be updated towards the distribution of new soft Q-function. Moreover, these three optimizations are done by stochastic gradient descent with Adam optimizer and fixed learning rates. The gradients of the objective Functions (6)–(8) are defined as follows:

$$\nabla_\psi J_V(\psi) = \mathbb{E}_{q_t}[\frac{1}{2}(\nabla_\psi V_\psi(q_t||q_{\text{goal}}) - \mathbb{E}_{a_t}[\min_{k=1,2} Q_{\theta_k}(q_t||q_{\text{goal}}, a_t) - \beta \log \pi_\phi(a_t|\, q_t||q_{\text{goal}})])^2], \quad \text{(A2)}$$

$$\nabla_{\theta_{k=1,2}} J_Q(\theta_{k=1,2}) = \mathbb{E}_{q_t,a_t}[\frac{1}{2}(\nabla_{\theta_{k=1,2}} Q_{\theta_{k=1,2}}(q_t||q_{\text{goal}}, a_t) - (r_{t+1} + V_{\bar{\psi}}(q_{t+1}||q_{\text{goal}})))^2], \quad \text{(A3)}$$

$$\nabla_\phi J_\pi(\phi) = \mathbb{E}_{q_t,a_t}[\beta \nabla_\phi \log \pi_\phi(a_t|q_t||q_{\text{goal}}) - \nabla_\phi \min_{k=1,2} Q_{\theta_k}(q_t||q_{\text{goal}}, a_t)] \quad \text{(A4)}$$

$$\approx \beta \nabla_\phi \log \pi_\phi(a_t|q_t||q_{\text{goal}}) + (\beta \nabla_{a_t} log \pi_\phi(a_t|q_t||q_{\text{goal}}) - \nabla_{a_t} \min_{k=1,2} Q_{\theta_k}(q_t||q_{\text{goal}}, a_t)) \nabla_\phi f_\phi(\epsilon_t, q_t||q_{\text{goal}}). \quad \text{(A5)}$$

Note that Equation (A5) is an approximated and decomposed form of Equation (A4) where $a_t = f_\phi(\epsilon_t, q_t||q_{\text{goal}})$. These three gradients are used to solve each of the minimization problems. The complete algorithm is described in Algorithm 1.

Algorithm 1 Proposed SAC-based path planning algorithm for multi-arm manipulator.

```
Define MAMMDP and the augmented state $q_t$ and the state and goal state $q_{\text{init}}$ and $q_{\text{goal}}$
Initialize network parameters $\psi, \theta_{1,2}, \phi$
Initialize the parameter values of the target network $\bar{\psi} \leftarrow \psi$
Initialize global replay memory $\mathcal{D}$

for $e = 1$ to $M$ do
    Initialize local buffer $\mathcal{L}$                                    ▷ Memory for an episode
    for $t = 0$ to $T - 1$ do
        Randomly choose the goal and initial positions $q_{\text{goal}}, q_{\text{init}} \in \mathbf{Q}^a_{\text{free}}$
        $a_t = f_\phi(\epsilon_t, q_t || q_{\text{goal}}),\ \epsilon_t \sim \mathcal{N}(0, \sigma_t)$
        $\hat{q}_{t+1} = q_t + \alpha \cdot a_t + \epsilon_e,\ \ \epsilon_e \sim \mathcal{N}(0, \sigma_e)$

        if $\hat{q}_{t+1} \in \mathbf{Q}^a_{\text{free}}$ then                              ▷ Get next state and reward
            $q_{t+1} \leftarrow \hat{q}_{t+1}$
            $r_{t+1} = -1$
        else if $\hat{q}_{t+1} \in \mathbf{Q}^a_{\text{collide}}$ then
            $q_{t+1} \leftarrow q_t$
            $r_{t+1} = -1$
        else if $|q_{t+1} - q_{\text{goal}}| \leq \eta \cdot \alpha$ then
            $r_{t+1} = 0$
            Terminate due to goal arrival
        end if

        Store the transition $(q_t || q_{\text{goal}}, a_t, r_{t+1}, q_{t+1} || q_{\text{goal}})$ in $\mathcal{D}, \mathcal{L}$
                                                                              ▷ Parameters update
        Sample mini-batch of $m$ transitions $(q_l || q_{\text{goal}}, a_l, r_{l+1}, q_{l+1} || q_{\text{goal}})$ from $\mathcal{D}$
            $J_V(\psi) = \mathbb{E}_{q_l}[\frac{1}{2}(V_\psi(q_l || q_{\text{goal}}) - \mathbb{E}_{a_l}[\min_{k=1,2} Q_{\theta_k}(q_l || q_{\text{goal}}, a_l) - \beta \log \pi_\phi(a_l |\ q_l || q_{\text{goal}})])^2]$
            $J_Q(\theta_{k=1,2}) = \mathbb{E}_{q_l, a_l}[\frac{1}{2}(Q_{\theta_{k=1,2}}(q_l || q_{\text{goal}}, a_l) - (r_{l+1} + V_{\bar{\psi}}(q_{l+1} || q_{\text{goal}})))^2]$
            $J_\pi(\phi) = \mathbb{E}_{q_l, a_l}[\beta \log \pi_\phi(a_l | q_l || q_{\text{goal}}) - \min_{k=1,2} Q_{\theta_k}(q_l || q_{\text{goal}}, a_l)]$

        Each network parameters $\psi, \theta_{1,2}, \phi$ are updated by gradient descent
        using $\nabla_\psi J_V(\psi), \nabla_{\theta_1} J_Q(\theta_1), \nabla_{\theta_2} J_Q(\theta_2), \nabla_\phi J_\pi(\phi)$

        Update state value target $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
    end for

    if $q_T \neq q_{\text{goal}}$ then                                         ▷ HER
        Set additional goal $q'_{\text{goal}} \in \{q_1, q_2, \cdots, q_T\}$
        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 $\mathcal{L}$
            if $|q_{t+1} - q'_{\text{goal}}| \leq \eta \cdot \alpha$ then
                $r'_{t+1} = 0$
            else $r'_{t+1} = -1$
            end if
            Store the transition $(q_t || q'_{\text{goal}}, a_t, r'_{t+1}, q_{t+1} || q'_{\text{goal}})$ in $\mathcal{D}$
        end for
    end if
end for
```

References

1. Spong, M.; Hutchinson, S.; Vidyasagar, M. *Robot Modeling and Control*; Wiley: New York, NY, USA, 2006.
2. Albu-Schäffer, A.; Hirzinger, G. A globally stable state feedback controller for flexible joint robots. *Adv. Robot.* **2001**, *15*, 799–814. [CrossRef]
3. Basile, F.; Caccavale, F.; Chiacchio, P.; Coppola, J.; Curatella, C. Task-oriented motion planning for multi-arm robotic systems. *Robot. -Comput.-Integr. Manuf.* **2012**, *28*, 569–582. [CrossRef]
4. Shome, R.; Bekris, K.E. Anytime Multi-arm Task and Motion Planning for Pick-and-Place of Individual Objects via Handoffs. In Proceedings of the 2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), New Brunswick, NJ, USA, 22–23 August 2019; pp. 37–43.
5. Lynch, K.M.; Park, F.C. *Modern Robotics: Mechanics, Planning, and Control*; Cambridge University Press: Cambridge, England, 2017.
6. Choset, H.M.; Hutchinson, S.; Lynch, K.M.; Kantor, G.; Burgard, W.; Kavraki, L.E.; Thrun, S.; Arkin, R.C. *Principles of Robot Motion: Theory, Algorithms, and Implementation*; MIT Press: Cambridge, MA, USA, 2005.
7. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
8. 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]
9. Gharbi, M.; Cortés, J.; Simeon, T. A sampling-based path planner for dual-arm manipulation. In Proceedings of the 2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Xi'an, China, 2–5 July 2008; pp. 383–388.
10. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; Volume 2, pp. 995–1001.
11. LaValle, S.M.; Kuffner, J.J. Rapidly-exploring random trees: Progress and prospects. *Algorithmic Comput. Robot. New Dir.* **2001**, *5*, 293–308.
12. Preda, N.; Manurung, A.; Lambercy, O.; Gassert, R.; Bonfè, M. Motion planning for a multi-arm surgical robot using both sampling-based algorithms and motion primitives. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 1422–1427.
13. Kurosu, J.; Yorozu, A.; Takahashi, M. Simultaneous Dual-Arm Motion Planning for Minimizing Operation Time. *Appl. Sci.* **2017**, *7*, 1210. [CrossRef]
14. Kavraki, L.E.; Latombe, J.C.; Motwani, R.; Raghavan, P. Randomized Query Processing in Robot Path Planning. *J. Comput. Syst. Sci.* **1995**, *57*, 50–60. [CrossRef]
15. Hsu, D.; Latombe, J.C.; Kurniawati, H. On the Probabilistic Foundations of Probabilistic Roadmap Planning. *Int. J. Robot. Res.* **2006**, *25*, 627–643. [CrossRef]
16. De Santis, A.; Albu-Schaffer, A.; Ott, C.; Siciliano, B.; Hirzinger, G. The skeleton algorithm for self-collision avoidance of a humanoid manipulator. In Proceedings of the IEEE/ASME international conference on advanced intelligent mechatronics, Zurich, Switzerland, 4–7 September 2007; pp. 1–6.
17. Dietrich, A.; Wimböck, T.; Täubig, H.; Albu-Schäffer, A.; Hirzinger, G. Extensions to reactive self-collision avoidance for torque and position controlled humanoids. In Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 3455–3462.
18. Sugiura, H.; Gienger, M.; Janssen, H.; Goerick, C. Real-Time Self Collision Avoidance for Humanoids by means of Nullspace Criteria and Task Intervals. In Proceedings of the 6th IEEE-RAS International Conference on Humanoid Robots, Genova, Italy, 4–6 December 2006; pp. 575–580.
19. Martínez, C.; Jiménez, F. Implementation of a Potential Field-Based Decision-Making Algorithm on Autonomous Vehicles for Driving in Complex Environments. *Sensors* **2019**, *19*, 3318. [CrossRef]
20. Sichkar, V.N. Reinforcement Learning Algorithms in Global Path Planning for Mobile Robot. In Proceedings of the International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russia, 25–29 March 2019; pp. 1–5.
21. Wang, C.; Zhang, Q.; Tian, Q.; Li, S.; Wang, X.; Lane, D.; Petillot, Y.; Wang, S. Learning Mobile Manipulation through Deep Reinforcement Learning. *Sensors* **2020**, *20*, 939. [CrossRef]

22. Guo, S.; Zhang, X.; Zheng, Y.; Du, Y. An Autonomous Path Planning Model for Unmanned Ships Based on Deep Reinforcement Learning. *Sensors* **2020**, *20*, 426. [CrossRef]
23. Bae, H.; Kim, G.; Kim, J.; Qian, D.; Lee, S. Multi-Robot Path Planning Method Using Reinforcement Learning. *Appl. Sci.* **2019**, *9*, 3057. [CrossRef]
24. Gu, S.; Holly, E.; Lillicrap, T.; Levine, S. Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3389–3396.
25. Liu, C.; Gao, J.; Bi, Y.; Shi, X.; Tian, D. A Multitasking-Oriented Robot Arm Motion Planning Scheme Based on Deep Reinforcement Learning and Twin Synchro-Control. *Sensors* **2020**, *20*, 3515. [CrossRef] [PubMed]
26. Umay, I.; Fidan, B.; Melek, W. An Integrated Task and Motion Planning Technique for Multi-Robot-Systems. In Proceedings of the IEEE International Symposium on Robotic and Sensors Environments (ROSE), Ottawa, ON, Canada, 17–18 June 2019; pp. 1–7.
27. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 15 July 2018; pp. 1861–1870.
28. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; Zaremba, W. Hindsight Experience Replay. In Proceedings of the 31st Conference on Neural Information Processing System, Long Beach, CA, USA, 4–9 December 2017; pp. 5048–5058.
29. Chen, J.; Zhou, Y.; Gong, J.; Deng, Y. An Improved Probabilistic Roadmap Algorithm with Potential Field Function for Path Planning of Quadrotor. In Proceedings of the Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 3248–3253.
30. Kim, M.; Han, D.K.; Park, J.H.; Kim, J.S. Motion Planning of Robot Manipulators for a Smoother Path Using a Twin Delayed Deep Deterministic Policy Gradient with Hindsight Experience Replay. *Appl. Sci.* **2020**, *10*, 575. [CrossRef]
31. Latombe, J.C. *Robot Motion Planning*; Kluwer Academic Publishers: Norwell, MA, USA, 1991.
32. Lozano-Perez. Spatial Planning: A Configuration Space Approach. *IEEE Trans. Comput.* **1983**, *C-32*, 108–120. [CrossRef]
33. Laumond, J.P.P. Robot Motion Planning and Control. In *Lecture Notes in Control and Information Sciences*; Springer: Berlin, Germany, 1998.
34. Bergen, G.V.D.; Bergen, G.J. *Collision Detection in Interactive 3D Environments*, 1st ed.; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2003. [CrossRef]
35. Ericson, C. *Real-Time Collision Detection*; CRC Press, Inc.: Boca Raton, FL, USA, 2004. [CrossRef]
36. Fares, C.; Hamam, Y. Collision Detection for Rigid Bodies: A State of the Art Review. GraphiCon 2005. Available online: https://https://www.graphicon.org/html/2005/proceedings/papers/Fares.pdf (accessed on 19 August 2019)
37. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed.; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1994.
38. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018; ISBN 978-0-262-03924-6.
39. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In Proceedings of the 12th International Conference on Neural Information Processing Systems, Denver, CO, USA, 30 November–2 December 1999; NIPS'99, pp. 1057–1063.
40. 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, Beijing, China, 22–24 June 2014; pp. I–387–I–395.
41. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. ICLR (Poster). *arXiv* **2016**, arXiv:1509.02971.
42. Abdolmaleki, A.; Springenberg, J.T.; Tassa, Y.; Munos, R.; Heess, N.; Riedmiller, M. Maximum a Posteriori Policy Optimisation. *arXiv* **2018**, arXiv:1806.06920.
43. Barth-Maron, G.; Hoffman, M.W.; Budden, D.; Dabney, W.; Horgan, D.; Dhruva, T.; Muldal, A.; Heess, N.; Lillicrap, T. Distributed Distributional Deterministic Policy Gradients. *arXiv* **2018**, arXiv:1804.08617.
44. Degris, T.; White, M.; Sutton, R. Off-Policy Actor-Critic. *arXiv* **2012**, arXiv:1205.4839.

45. 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* **2015**, *518*, 529–533. [CrossRef]
46. Fujimoto, S.; Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv* **2018**, arXiv:1802.09477.
47. Haarnoja, T.; Tang, H.; Abbeel, P.; Levine, S. Reinforcement Learning with Deep Energy-Based Policies. *arXiv* **2017**, arXiv:1702.08165.
48. Hasselt, H.V. Double Q-learning. *Adv. Neural Inf. Process. Syst.* **2010**, 2613–2621. Available online: http://papers.nips.cc/paper/3964-double-q-learning (accessed on 19 August 2019).
49. Green, P.J.; Silverman, B.W. *Nonparametric Regression and Generalized Linear Models: a Roughness Penalty Approach*; CRC Press: Boca Raton, FL, USA, 1993.