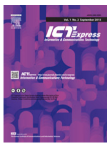
The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express 9 (2023) 403–408



www.elsevier.com/locate/icte

Deep reinforcement learning-based model-free path planning and collision avoidance for UAVs: A soft actor–critic with hindsight experience replay approach

Myoung Hoon Lee, Jun Moon[*]

Research Institute of Electrical and Computer Engineering, Hanyang University, Seoul 04763, South Korea
Department of Electrical Engineering, Hanyang University, Seoul 04763, South Korea

Received 17 November 2021; received in revised form 7 May 2022; accepted 7 June 2022
Available online 14 June 2022

Abstract

In this paper, we propose a soft actor–critic (SAC) algorithm with hindsight experience replay (HER), called SACHER, which is a class of deep reinforcement learning (DRL) algorithm. SAC is an off-policy model-free DRL algorithm that outperforms earlier DRL algorithms in terms of exploration and robustness. However, in SAC, maximizing the entropy-augmented objective degrades the optimality of learning outcomes. We propose SACHER to improve the learning performance of SAC. We apply SACHER to the path planning and collision avoidance control of unmanned aerial vehicles (UAVs). We demonstrate the effectiveness of SACHER in terms of the success rate, learning speed, and collision avoidance performance of UAV operation.

Keywords: Deep reinforcement learning; Soft actor–critic; Hindsight experience replay; UAV path planning; Collision avoidance and control

1. Introduction

In recent years, problems in the navigation and control of unmanned aerial vehicles (UAVs) have been studied for various applications such as target tracking, formation, and collision avoidance [1,2]. Most recent studies on the navigation and control of UAVs depend on the model accuracy and/or prior knowledge regarding the operational environment. However, collecting accurate model and operation information is challenging due to the lack of complete environmental information. Deep reinforcement learning (DRL) is an alternative approach to overcome these limitations, because it does not require any UAV model information and can be applied in various operational environments [3,4].

Soft actor–critic (SAC) is an off-policy DRL algorithm that optimizes stochastic policy based on the maximum entropy framework [5]. SAC is able to handle large continuous state and action spaces, and has advantages in terms of exploration and robustness compared to other DRL algorithms. In other words, SAC outperforms earlier DRL methods such as deep deterministic policy gradient (DDPG) [6], twin delayed deep deterministic policy gradient algorithm [7], and proximal policy optimization [8] in terms of learning speed and cumulative reward [5]. Note that although SAC can learn various environments with advantages in terms of exploration and robustness, the maximum entropy framework in SAC may degrade the optimality of learning outcomes after reaching the steady-state phase.

Recently, hindsight experience replay (HER) was proposed in [9] to improve the learning performance of DDPG. Specifically, HER is a sample-efficient experience replay method that enhances the performance of off-policy DRL algorithms by allowing the DRL agent to learn from both failures and successes, similar to humans. By using the concept of *goals*, HER provides a supplementary reward to the DRL agent, which improves the optimality of learning outcomes even if the goal is not achieved. Note that although DDPG with HER can deal with environments with large continuous state and action spaces, it still suffers from instability, i.e., it may converge to unstable solutions or diverge as a result of its high sensitivity to hyperparameters [10].

[*] Corresponding author at: Department of Electrical Engineering, Hanyang University, Seoul 04763, South Korea.
E-mail addresses: leemh91@hanyang.ac.kr (M.H. Lee), junmoon@hanyang.ac.kr (J. Moon).

Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2022.06.004
2405-9595/© 2022 The Authors. Published by Elsevier B.V. on behalf of The Korean Institute of Communications and Information Sciences. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

In this paper, we propose a DRL algorithm called the soft actor–critic (SAC) with hindsight experience replay (HER) (SACHER). As mentioned previously, SAC outperforms earlier DRL algorithms in terms of exploration, robustness, and learning performance. However, in SAC, maximizing the entropy-augmented objective function may degrade the optimality of learning outcomes. We resolve this limitation by proposing SACHER, which improves the learning performance of SAC using HER. More precisely, SACHER achieves the desired optimal outcomes faster and more accurately than SAC because HER improves the sample efficiency of SAC. Additionally, SACHER is able to avoid the instability observed in DDPG with HER.

We apply SACHER to the path planning and collision avoidance control problem of UAVs, where SACHER generates the optimal navigation path for UAVs under various obstacles. Based on simulation benchmark results, we demonstrate the effectiveness of SACHER in terms of the success rate, learning speed, and collision avoidance performance of UAV operation. Note that in UAV path planning and collision avoidance control problems, SACHER can be applied to arbitrary models of UAVs because it does not require specific information regarding UAV models and types of controllers.

In summary, the main contributions of the paper can be stated as follows:

(a) We apply HER to SAC and propose SACHER to improve the learning performance of SAC;
(b) We apply SACHER to the design of a model-free path planning and collision avoidance control system for UAVs considering various obstacles.

It should be mentioned that [11,12] also studied the path planning and control of robot manipulators using a technique similar to that used in this paper. However, [11] requires specific assumptions in which the system must be fully actuated and the associated Markov decision process (MDP) must be designed for a specific robot manipulator. Moreover, [12] can be applied to only specific types of sparse rewards and did not provide a detailed analysis of the implementation and application of the corresponding algorithm to the system. Unlike the method from [11], the proposed SACHER can be applied to under-actuated systems (e.g., UAVs) and uses the general MDP framework. In addition, differently from [12], the proposed SACHER considers both sparse and shaped rewards and we provide a detailed analysis of the implementation of the SACHER in a UAV system. We note that the problem setup, approaches used, and main results of this study are completely different from those of [11,12].

The remainder of this paper is organized as follows. SACHER is presented in Section 2. The simulation setup and environmental design of the SACHER-based UAV control method are presented in Section 3. The simulation results are presented in Section 4. We conclude this paper in Section 5.

2. SACHER: Soft Actor–Critic algorithm with Hindsight Experience Replay

In this section, we first describe SAC [5] and HER [9]. We then present SACHER and explain in detail.

2.1. Soft Actor–Critic (SAC) algorithm

As studied in [5], SAC is a class of the maximum entropy DRL algorithm that optimizes the following objective function:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))\right], \tag{1}$$

where $r_t = r(s_t, a_t)$ is the reward obtained when the agent executes the action $a_t \in \mathcal{A}$ in the state $s_t \in \mathcal{S}$, π is the policy, ρ_π is the joint distribution over states and actions induced by the policy π, α is the temperature weight of the entropy term $\mathcal{H}$, and $\mathcal{H}(\pi(\cdot|s_t)) = -\mathbb{E}_\pi[\log\pi(\cdot|s_t)] = -\int_{\mathcal{A}}\pi(a|s_t)\log\pi(a|s_t)da$ is the entropy. Here, $\mathcal{A}$ and $\mathcal{S}$ denote action and state spaces, respectively.

The main objective of SAC is to find the optimal policy π^* that maximizes the entropy-augmented objective function $J(\pi)$ in (1), which requires *soft policy iteration* on soft Q-functions and state value functions. The soft Q-function satisfies the following soft Bellman equation:

$$Q(s_t,a_t) := r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[V(s_{t+1})\right], \tag{2}$$

where $V(s_t) := \mathbb{E}_{a_t\sim\pi}\left[Q(s_t,a_t) - \alpha\log\pi(a_t|s_t)\right]$ is the soft state value function, and $p = p(s_{t+1}|a_t,s_t)$ is the state transition probability, which represents the probability density of the next state $s_{t+1} \in \mathcal{S}$ given the current state $s_t \in \mathcal{S}$ and the action $a_t \in \mathcal{A}$. We can evaluate the soft Q-value of a fixed policy π by applying the Bellman equation in (2) to each time step, which is so-called *soft policy evaluation*.

The objective function for updating the policy can be written as follows:

$$J_\pi(\pi) = \mathbb{E}_{s_t\sim p}\left[D_{KL}\left(\pi(\cdot|s_t)\,\middle\|\,\frac{\exp\left(\frac{1}{\alpha}Q(s_t,\cdot)\right)}{Z(s_t)}\right)\right], \tag{3}$$

where D_{KL} denotes the Kullback–Leibler (KL) divergence, and $Z(s_t)$ is the partition function used to normalize the distribution. The minimization of the objective J_π in (3) with respect to the policy π is called *soft policy improvement*. We can easily verify that the original maximization problem of (1) is equivalent to the minimization of (3) because of the definition of D_{KL} [5].

Repeating soft policy evaluation and soft policy improvement is called *soft policy iteration*. SAC can find the optimal policy π^* and the corresponding optimal Q-value through soft policy iteration. Note that applying soft policy iteration directly to environments having large continuous state and action spaces requires a certain type of practical approximation [5].

Instead of executing the policy iteration until convergence, parameterized neural networks for the Q-function and policy are used as function approximators. The soft Q-network is parameterized by θ, and the parameter for the soft policy network is denoted by ϕ. The soft Q-function parameters θ are optimized by minimizing the squared soft Bellman residual given by

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\Big[\frac{1}{2}(Q_\theta(s_t,a_t) - (r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}}V_{\bar{\theta}}(s_{t+1})))^2\Big], \tag{4}$$

where $\mathcal{D}$ denotes the replay buffer, and $\bar{\theta}$ is the target Q-function parameter. Note that the soft state value function V is also parameterized by the soft Q-function parameter θ according to its relationship with the Q-function in (2). Soft policy parameters ϕ are learned by minimizing the expected KL divergence in (3):

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}}\left[\mathbb{E}_{a_t\sim\pi_\phi}\left[\alpha \log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t)\right]\right]. \tag{5}$$

Finally, the soft Q-networks, soft policy network, and temperature weight α are optimized using the stochastic gradient descent (SGD) method. For SGD, we use two soft Q-networks parameterized by θ_j, $j = 1, 2$, which are trained independently to optimize the soft Bellman residual in (4). The minimum of the two soft Q-functions is used for SGD to minimize the loss functions in (4) and (5). In each gradient step of SGD, the Q-function parameters θ_1 and θ_2, and policy parameters ϕ are optimized to minimize the loss functions in (4) and (5), respectively. The temperature weight α is optimized in each gradient step of SGD to minimize the following objective:

$$J(\alpha) = \mathbb{E}_{a_t\sim\pi}\left[-\alpha \log \pi(a_t|s_t) - \alpha\bar{\mathcal{H}}\right], \tag{6}$$

where $\bar{\mathcal{H}}$ denotes the desired target entropy. The target Q-function parameters $\bar{\theta}_1$ and $\bar{\theta}_2$ are updated using the exponential moving average method with the smoothing constant δ. After sufficient iterations of the learning process, SACHER gives the optimized soft Q-function parameters θ_1 and θ_2, and the soft policy parameters ϕ.

2.2. Hindsight Experience Replay (HER)

The main idea of hindsight experience replay (HER) in [9] is to allow DRL agents to learn from both failures and successes, similar to humans. To achieve this, HER employs the concept of a *goal* $g \in \mathcal{G}$, which was used in [13], where g represents the goal (or objective) that the DRL agent has to achieve in the environment and $\mathcal{G}$ represents the corresponding goal space. Then the modified reward function $r_t = r(s_t, a_t, g)$ is defined as a function of not only the state and action, but also the goal. The closer the state s_t is to the goal g, the greater the reward the DRL agent receives.

The detailed process of HER is as follows. First, we initialize an off-policy DRL algorithm $\mathbb{A}$ and empty the replay buffer $\mathcal{D}$. In each episode, an initial state s_0 and a goal g are uniformly sampled from the state space $\mathcal{S}$ and the goal space $\mathcal{G}$, respectively. Then, the DRL algorithm $\mathbb{A}$ interacts with the environment during environment steps $t = 1, 2, \ldots, T$ to obtain the transition tuple $(s_t, a_t, r_t, s_{t+1}, g)$. After executing the environment steps, HER has knowledge regarding the visited states $\zeta = \{s_0, s_1, \ldots, s_T\}$. Based on this knowledge, HER stores every transition tuple (s_t, a_t, r_t, s_{t+1}) together with the original goal g in the replay buffer $\mathcal{D}$. HER then stores extra transition tuples $(s_t, a_t, r'_t, s_{t+1})$ together with $g' \in \varphi$, where $\varphi = \{g'_1, g'_2, \ldots, g'_m\}$ is a set of m additional goals uniformly sampled from the visited states $\zeta = \{s_0, s_1, \ldots, s_T\}$. Through this process, HER provides supplementary rewards $r'_t = r(s_t, a_t, g')$ to the DRL agent even if the goal g is not achieved. This process enhances DRL algorithms in terms of learning speed and the success rate of reaching the goal.

Algorithm 1 SACHER

```
Input: $\theta_1$, $\theta_2$, $\phi$
  $\bar{\theta}_1 \leftarrow \theta_1$,$\bar{\theta}_2 \leftarrow \theta_2$, $\mathcal{D} \leftarrow \emptyset$
  for episode $k = 1, K$ do
    Sample an initial state $s_0 \in \mathcal{S}$ and a goal $g \in \mathcal{G}$
    for environment step $t = 0, T$ do
      $a_t \sim \pi_\phi(a_t|s_t, g)$
      $s_{t+1} \sim p(s_{t+1}|a_t, s_t, g)$
    end for
    for environment step $t = 0, T$ do
      $r_t := r(s_t, a_t, g)$
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1}, g)\}$
      Sample a set of additional goals $\varphi$ from $\zeta$
      for $g' \in \varphi$ do
        $r'_t = r(s_t, a_t, g')$
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r'_t, s_{t+1}, g')\}$
      end for
    end for
    for gradient step $n = 1, N$ do
      $\theta_j \leftarrow \theta_j - \lambda_Q \hat{\nabla}_{\theta_j} J_Q(\theta_j)$ for $j \in \{1, 2\}$
      $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
      $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J_\pi(\alpha)$
      $\bar{\theta}_j \leftarrow \delta\bar{\theta}_j + (1 - \delta)\bar{\theta}_j$ for $j \in \{1, 2\}$
    end for
  end for
Ouput: $\theta_1, \theta_2, \phi$
```

2.3. SACHER: Soft Actor–Critic algorithm with Hindsight Experience Replay

We apply HER to SAC and propose SACHER. The structure of SACHER is presented in Algorithm 1 and the detailed process of SACHER follows from Sections 2.1 and 2.2. In Algorithm 1, one remarkable change compared to Section 2.1 is that the entire framework of SACHER, including the policy π, transition probability p, reward r, and Q-function, becomes more complicated than that of SAC due to the implementation of the goal g in HER (see Algorithm 1). These complex procedures lead to intricacies when evaluating soft policy iteration as well as SGD. In the practical implementation of the SACHER algorithm, we resolve this difficulty by merging the goal g into the state s because the goal g is fixed and does not change during each training episode.

3. Simulation setup and environment design

In this section, we first describe the detailed simulation setup for the SACHER-based path planning and collision avoidance control problem for UAVs. Then, we design a UAV environment with obstacles for the SACHER agent to learn through interaction.

3.1. SACHER-based UAV path navigation and control

The entire framework of the SACHER-based UAV path planning and collision avoidance control system is illustrated

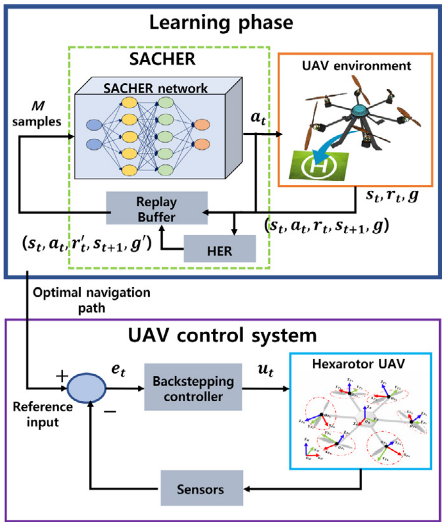


Fig. 1. Framework for SACHER-based UAV path planning and collision avoidance control.

in Fig. 1. In the learning phase, SACHER learns the UAV environment and then generates an optimal navigation path for UAVs using Algorithm 1. After completing the learning phase, the output of SACHER is considered as the reference path for the UAV control system. Hence, SACHER can be regarded as a path planning system for UAVs. In the UAV control system, the tracking controller controls UAVs to follow the optimal navigation path generated by SACHER.

In our simulations, the tilted-hexarotor UAV model in [14] is adopted. Additionally, we design the standard backstepping controller (see [15]) for the hexarotor to track the optimal navigation path generated by SACHER. Note that in the learning phase, because SACHER is model-free and does not require any information regarding UAV dynamics models (e.g., quadrotor, aircraft, or ground vehicles) or types of controllers, any (optimal/nonoptimal) nonlinear (or linear) controllers (with appropriate design modifications) can be used, as shown in Fig. 1, instead of the hexarotor and backstepping controller.

3.2. Environment design: UAV collision avoidance under obstacles

We consider a complex UAV operation in which the UAV seeks to land in a given landing area (goal) by following the shortest path while avoiding obstacles. To design this environment, we use a simple position and angle updating equation for the UAV. Specifically, let (x, y, z) be the position coordinate in three-dimensional space, ψ and $\dot{\psi}$ be the yaw angle and yaw angular speed, respectively, and τ be the yaw torque of the UAV. Then, the position and angle updating equation for the UAV can be written as follows:

$$\begin{cases} x_{t+1} = x_t + v_1 \cos(\psi_t)\Delta t, & \psi_{t+1} = \psi_t + \dot{\psi}_t \Delta t, \\ y_{t+1} = y_t + v_1 \sin(\psi_t)\Delta t, & \dot{\psi}_{t+1} = \dot{\psi}_t + \tau_t \Delta t, \\ z_{t+1} = z_t - v_2 \Delta t, & \end{cases} \tag{7}$$

Table 1
SACHER hyperparameters.

Hyperparameter	Value
Learning rate for optimizer $\lambda_Q, \lambda_\pi, \lambda$	3×10^{-4}
Discount factor γ	0.99
Number of hidden layers	2
Number of hidden units	256
Minibatch size	256
Target smoothing coefficient δ	0.005
Activation function	ReLU
Replay buffer capacity	10^6
Number of additional goals (HER) m	4
Target entropy $\bar{\mathcal{H}}$	-1

where v_1 and v_2 are the updating rates for path generation and Δt is the sampling time. While interacting with the environment, the SACHER agent observes the updated state $s = [x, y, z, \psi, \dot{\psi}]^\top$ in (7) and uses the yaw torque τ as an action ($a = \tau$).

The main objective of UAV path planning and collision avoidance control is to reach the landing area on the xy-plane. We define $g = [g_x, g_y]^\top$ as the center of the landing area, which is the goal in the UAV environment. Because the goal g is defined on the xy-plane, whether the goal is achieved depends on states x and y, which are updated by (7).

Next, we consider N cylindrical obstacles in the UAV environment. An individual cylindrical obstacle $\mathcal{O}_i$, $1 \leq i \leq N$, is represented by the following zero-sublevel set:

$$\mathcal{O}_i := \{h \in \mathbb{R}^3 \mid (x - x_{o,i})^2 + (y - y_{o,i})^2 - r_{o,i}^2 \leq 0\}, \tag{8}$$

where $h = [x \ y \ z]^\top$ is the location of the obstacle in xyz-space, $r_{o,i}$ is the radius of the obstacle, and $(x_{o,i}, y_{o,i})$ is the center of the obstacle on the xy-plane.

The reward for each environmental step is defined as

$$r(s_t, g) = \bar{r}(s_t, g) + \sum_{i=1}^{N} p_i(s_t), \tag{9}$$

where $\bar{r} = -k_1 \left((x_t - g_x)^2 + (y_t - g_y)^2\right) - k_2(z_t)^2$ and $p_i = -c_1$ if $(x_t - x_{o,i})^2 + (y_t - y_{o,i})^2 - r_{o,i}^2 \leq c_2$ (otherwise, $p_i = 0$) with positive constant weights k_1 and k_2 and penalty constants c_1 and c_2. The penalty constant c_1 decreases the reward in (9) when the UAV collides with obstacles in (8). The other penalty constant c_2 acts as a margin that prevents the UAV from colliding with obstacles. Under the UAV environment designed in (7)–(9), SACHER seeks to find the optimal path to reach the landing area without collisions by maximizing the cumulative reward $R = \sum_{t=0}^{T} r_t$.

4. Simulation results

We provide the simulation results for the SACHER-based path planning and collision avoidance control problem for the hexarotor UAV. The detailed simulation setup is described in Section 3 and Fig. 1. To compare and validate the learning performance of SACHER, we also provide the simulation results of SAC [5] and DDPG [6] under the same simulation settings.

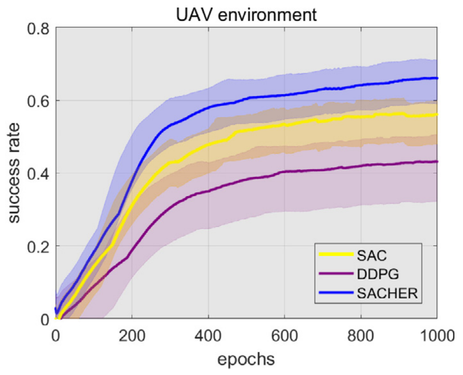


Fig. 2. Success rate curves of SAC, DDPG and SACHER in the UAV environment. The solid curve is the mean and the shaded area represents the minimum and maximum success rates in four trials.

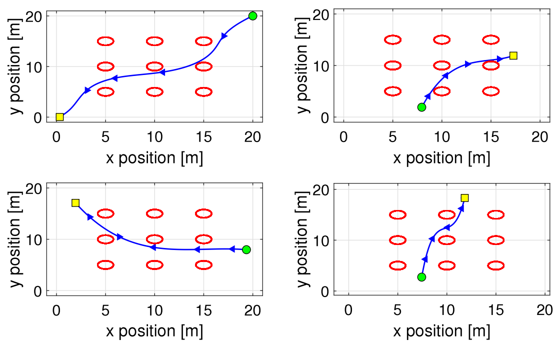


Fig. 3. Four optimal navigation paths of UAV generated by SACHER in the UAV environment. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The values of SACHER hyperparameters are listed in Table 1. The values of the UAV environment parameters in (7) and (9) are $v_1 = 2$, $v_2 = 0.5$, $\Delta t = 0.1$, $\tau_t \in [-0.5, 0.5]$, $k_1 = 10^{-3}$, $k_2 = 10^{-4}$, $c_1 = 10$, and $c_2 = 0.2$. For (8), we consider a case where there are $N = 9$ obstacles with locations $(x_{o,i}, y_{o,i})$ of $\{(5, 5), (5, 10), (5, 15), (10, 5), \ldots, (15, 10), (15, 15)\}$ with the same radius of $r_{o,i} = 1$.

Fig. 2 shows the success rate curves of SAC, DDPG, and SACHER in the UAV environment during the learning stage. We train each algorithm for 1000 epochs, and every epoch includes 50 episodes. Each episode carries out 200 environmental steps. The results in Fig. 2 show that SACHER outperforms SAC and DDPG in terms of learning speed and performance. After sufficient learning, SACHER successfully generates optimal paths for the UAV from random initial positions to random goals with a 66.08% success rate, whereas those of SAC and DDPG are only 56.15% and 43.30%, respectively.

Fig. 3 shows four optimal paths generated by SACHER for the UAV from random initial positions to random goals with nine obstacles. The blue-colored line is the optimal path of the UAV generated by SACHER, the green-colored circle is the initial position of the UAV, the yellow-colored square is the goal (landing area), and the red-colored circles represent the locations of nine cylindrical obstacles. From Fig. 3, we can see that SACHER generates various optimal paths for the UAV with randomly sampled initial position and goal pairs while avoiding collision with obstacles.

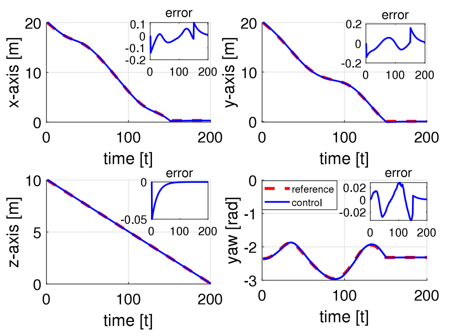


Fig. 4. Paths and tracking errors of the hexarotor UAV using the back-stepping controller in the UAV control system. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The simulation results for the UAV control system with the optimal path from SACHER are shown in Fig. 4. The paths of the positions and yaw angle, and their tracking errors are demonstrated in Fig. 4. The (red-colored) dotted line is the optimal navigation path generated by SACHER. The (blue-colored) solid line is the path of the hexarotor UAV controlled by the backstepping controller. From Fig. 4, it can be seen that the hexarotor UAV follows the optimal navigation path generated by SACHER with negligible tracking errors.

5. Conclusions

In this paper, we have proposed a DRL algorithm called SACHER. In SACHER, HER improves the sample efficiency and learning performance of SAC by allowing SAC to learn from both failures and successes when trying to achieve the desired goal. SACHER has been applied to the path planning and collision avoidance control problem of UAVs, where SACHER generates optimal navigation paths for UAVs. Note that SACHER for UAV navigation and control problems can be applied to arbitrary models of UAVs. The effectiveness of SACHER has been validated through simulations. One possible future work of this paper is to extend SACHER to partial observation cases and apply it to UAV control problems.

CRediT authorship contribution statement

Myoung Hoon Lee: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing, Visualization. **Jun Moon:** Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the AE and two anonymous referees for their careful reading and suggestions, which have helped to improve the paper.

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP), South Korea grant funded by the Korea government (MSIT) (No. 2020-0-01373, Artificial Intelligence Graduate School Program (Hanyang University), South Korea) and the research fund of Hanyang University, South Korea (HY-202100000320009).

References

[1] R. Wang, J. Liu, Trajectory tracking control of a 6-DOF quadrotor UAV with input saturation via backstepping, J. Franklin Inst. 355 (2018) 3288–3309.

[2] C.K. Verginis, Z. Xu, D.V. Dimarogonas, Decentralized motion planning with collision avoidance for a team of UAVs under high level goals, in: IEEE Int. Conf. Robot. Autom., 2017, pp. 781–787.

[3] W.J. Yun, S. Jung, J. Kim, J.-H. Kim, Distributed deep reinforcement learning for autonomous aerial eVTOL mobility in drone taxi applications, ICT Express 7 (1) (2021) 1–4.

[4] J. Moon, S. Papaioannou, C. Laoudias, P. Kolios, S. Kim, Deep reinforcement learning multi-UAV trajectory control for target tracking, IEEE Internet Things J. 8 (20) (2021) 15441–15455.

[5] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, S. Levine, Soft actor-critic algorithms and applications, 2018, arXiv:1812.05905.

[6] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, in: Int. Conf. Learn. Represent. (ICLR), 2016.

[7] S. Fujimoto, H. van Hoof, D. Meger, Addressing function approximation error in actor-critic methods, 2018, arXiv:1802.09477.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv:1707.06347.

[9] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, W. Zaremba, Hindsight experience replay, in: Adv. Neural Inf. Process. Syst. (NIPS), 2017.

[10] G. Matheron, N. Perrin, O. Sigaud, The problem with DDPG: understanding failures in deterministic environments with sparse rewards, 2019, arXiv:1911.11679.

[11] E. Prianto, M. Kim, J.-H. Park, J.-H. Bae, J.-S. Kim, Path planning for multi-arm manipulators using deep reinforcement learning: Soft actor–critic with hindsight experience replay, Sensors 20 (20) (2020) 1–22.

[12] T. Yan, W. Zhang, S.X. Yang, L. Yu, Soft actor-critic reinforcement learning for robotic manipulator with hindsight experience replay, Int. J. Robot. Autom. 34 (2019) http://dx.doi.org/10.2316/J.2019.206-0216.

[13] T. Schaul, D. Horgan, K. Gregor, D. Silver, Universal value function approximators, in: Int. Conf. Mach. Learn. (ICML), 2015.

[14] S. Rajappa, M. Ryll, H.H. Bülthoff, A. Franchi, Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers, in: IEEE Int. Conf. Robot. Autom., 2015, pp. 4006–4013.

[15] A. Isidori, Nonlinear Control Systems, Springer Science & Business Media, 2013.