A LiDAR Based Mobile Area Decision Method for TLS-DQN: Improving Control for AAV Mobility

Nobuki Saito[1], Tetsuya Oda[2(✉)], Aoto Hirata[1], Chihiro Yukawa[2], Elis Kulla[2], and Leonard Barolli[3]

[1] Graduate School of Engineering, Okayama University of Science (OUS), 1-1 Ridaicho, Kita-ku, Okayama 700-0005, Japan
{t21jm01md,t21jm02zr}@ous.jp
[2] Department of Information and Computer Engineering, Okayama University of Science (OUS), 1-1 Ridaicho, Kita-ku, Okayama 700-0005, Japan
oda@ice.ous.ac.jp, t18j097yc@ous.jp
[3] Department of Information and Communication Engineering, Fukuoka Institute d of Technology, 3-30-1 Wajiro-higashi, Higashi-ku, Fukuoka 811-0295, Japan
barolli@fit.ac.jp

Abstract. The Deep Q-Network (DQN) is one of the deep reinforcement learning algorithms, which uses deep neural network structure to estimate the Q-value in Q-learning. In the previous work, we designed and implemented a DQN-based Autonomous Aerial Vehicle (AAV) testbed and proposed a Tabu List Strategy based DQN (TLS-DQN). In this paper, we propose a LiDAR Based Mobile Area Decision Method for TLS-DQN to improve the control for AAV Mobility. The evaluation results show that the proposed method makes a good decision for the destination and mobile area based on LiDAR.

1 Introduction

The Unmanned Aerial Vehicle (UAV) is expected to be used in different fields such as aerial photography, transportation, search and rescue of humans, inspection, land surveying, observation and agriculture. Autonomous Aerial Vehicle (AAV) [1] has the ability to operate autonomously without human control and is expected to be used in a variety of fields, similar to UAV. So far many AAVs [2–4] are proposed and used practically. However, existing autonomous flight systems are designed for outdoor use and rely on location information by the Global Navigation Satellite System (GNSS) or others. On the other hand, in an environment where it is difficult to obtain position information from GNSS, it is necessary to determine a path without using position information. Therefore, autonomous movement control is essential to achieve operations that are independent of the external environment, including non-GNSS environments such as indoor, tunnel and underground.

L. Barolli (Ed.): 3PGCIC 2021, LNNS 343, pp. 30–42, 2022.
https://doi.org/10.1007/978-3-030-89899-1_4

In [5–8] the authors consider Wireless Sensor and Actuator Networks (WSANs), which can act autonomously for disaster monitoring. A WSAN consists of wireless network nodes, all of which have the ability to sense events (sensors) and perform actuation (actuators) based on the sensing data collected by the sensors. WSAN nodes in these applications are nodes with integrated sensors and actuators that have the high processing power, high communication capability, high battery capacity and may include other functions such as mobility. The application areas of WSAN include AAV [9], Autonomous Underwater Vehicle (AUV) [10], Autonomous Surface Vehicle (ASV) [11], Heating, Ventilation, Air Conditioning (HVAC) [12], Internet of Things (IoT) [13], Ambient Intelligence (AmI) [14], ubiquitous robotics [15], and so on.

Deep reinforcement learning [16] is an intelligent algorithm that is effective in controlling autonomous robots such as AAV. Deep reinforcement learning is an approximation method using deep neural network for value function and policy function in reinforcement learning. Deep Q-Network (DQN) is a method of deep reinforcement learning using Convolution Neural Network (CNN) as a function approximation of Q-values in the Q-learning algorithm [16,17]. DQN combines the neural fitting Q-iteration [18,19] and experience replay [20], shares the hidden layer of the action value function for each action pattern and can stabilize learning even with nonlinear functions such as CNN [21,22]. However, there are some points where learning is difficult to progress for problems with complex operations and rewards, or problems where it takes a long time to obtain a reward.

In this paper, we propose a LiDAR based mobile area decision method for TLS-DQN to improve the control for AAV mobility. Also, we present the simulation results for AAV control using TLS-DQN [23,24] and the proposed method. The structure of the paper is as follows. In Sect. 2, we show the DQN based AAV testbed. In Sect. 3, we describe the proposed method. In Sect. 4, we discuss the simulation results of TLS-DQN. Finally, conclusions and future work are given in Sect. 5.

2 DQN Based AAV Testbed

In this section, we discuss quadrotor for AAV and DQN for AAV mobility.

2.1 Quadrotor for AAV

For the design of AAV, we consider a quadrotor, which is a type of multicopter. Multicopter is high maneuverable and can operate in places that are difficult for people to enter, such as disaster areas and dangerous places. It also has the advantage of not requiring space for takeoffs and landings and being able to stop at mid-air during the flight, therefore enabling activities at fixed points. The quadrotor is a type of rotary-wing aircraft that uses four rotors for takeoff or propulsion, and can operate with less power than hexacopter and octocopter, and is less expensive to manufacture.

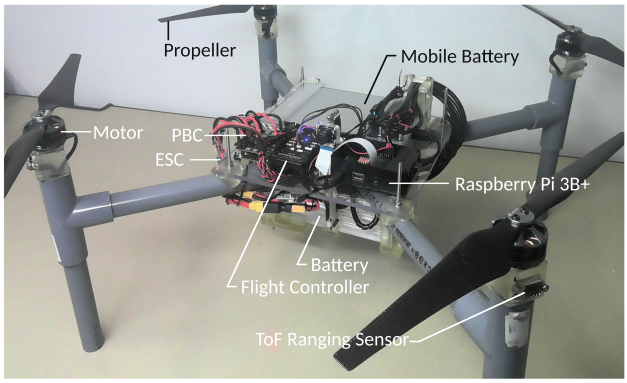


Fig. 1. Snapshot of AAV.

Table 1. Components of quadrotor.

Component	Model
Propeller	15 × 5.8
Motor	MN3508 700kv
Electric speed controller	F45A 32bitV2
Flight controller	Pixhawk 2.4.8
Power distribution board	MES-PDB-KIT
Li-Po battery	22.2v 12000mAh XT90
Mobile battery	Pilot Pro 2 23000mAh
ToF ranging sensor	VL53L0X
Raspberry Pi	3 Model B Plus
PVC pipe	VP20
Acrylic plate	5 mm

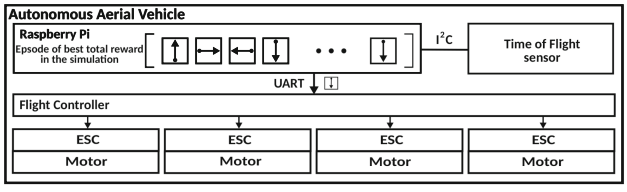


Fig. 2. AAV control system.

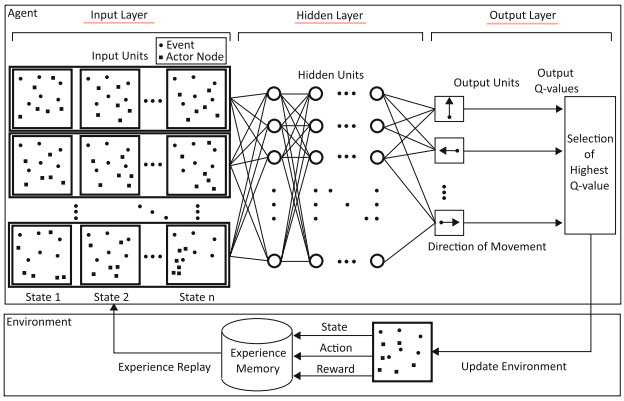


Fig. 3. DQN for AAV mobility control.

In Fig. 1 is shown a snapshot of the quadrotor used for designing and implementing an AAV testbed. The quadrotor frame is mainly composed of polyvinyl chloride (PVC) pipe and acrylic plate. The components for connecting the battery, motor, sensor, etc. to the frame are created using an optical 3D printer. Table 1 shows the components in the quadrotor. The size specifications of the quadrotor (including the propeller) are length 87 $[cm]$, width 87 $[cm]$, height 30 $[cm]$ and weight 4259 $[g]$.

In Fig. 2 is shown the AAV control system. The raspberry pi reads saved data of the best episode when carrying out the simulations by DQN and uses telemetry communication to send commands such as `up`, `down`, `forward`, `back`, `left`, `right` and `stop` to the flight controller. Also, multiple Time-of-Flight (ToF) range sensors using Inter-Integrated Circuit (I^2C) communication and General-Purpose Input Output (GPIO) are used to acquire and save flight data. The Flight Controller (FC) is a component that calculates the optimum motor rotation speed for flight based on the information sent from the built-in acceleration sensor and gyro sensor. The Electronic Speed Controller (ESC) is a part that controls the rotation speed of the motor in response to commands from FC. Through these sequences, AAV behaves and reproduces movement in simulation.

2.2 DQN for AAV Mobility

The DQN for moving control of AAV structure is shown in Fig. 3. The DQN for AAV mobility is implemented by Rust programming language [25].

In this work, we use the Deep Belief Network (DBN), because the computational complexity is smaller than CNN for DNN part in DQN. The environment is set as v_i. At each step, the agent selects an action a_t from the action sets of the mobile actuator nodes and observes a position v_t from the current state. The change of the mobile actuator node score r_t was regarded as the reward for the action. For the reinforcement learning, we can complete all of these mobile actuator nodes sequences m_t as Markov decision process directly, where sequences

of observations and actions are $m_t = v_1, a_1, v_2, \ldots, a_{t-1}, v_t$. A method known as experience replay is used to store the experiences of the agent at each timestep, $e_t = (m_t, a_t, r_t, m_{t+1})$ in a dataset $D = e_1, \ldots, e_N$, cached over many episodes into a Experience Memory. By defining the discounted reward for the future by a factor γ, the sum of the future reward until the end would be $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$. T means the termination time-step of the mobile actuator nodes. After running experience replay, the agent selects and executes an action according to an ϵ-greedy strategy. Since using histories of arbitrary length as inputs to a neural network can be difficult, Q-function instead works on fixed length format of histories produced by a function ϕ. The target was to maximize the action value function $Q^*(m, a) = \max_\pi E[R_t | m_t = m, a_t = a, \pi]$, where π is the strategy for selecting of best action. From the Bellman equation (see Eq. (1)), it is possibel to maximize the expected value of $r + \gamma Q^*(m', a')$, if the optimal value $Q^*(m', a')$ of the sequence at the next time step is known.

$$Q^*(m', a') = E_{m' \sim \xi}[r + \gamma_{a'} \max Q^*(m', a') | m, a]. \tag{1}$$

By not using iterative updating method to optimize the equation, it is common to estimate the equation by using a function approximator. Q-network in DQN is a neural network function approximator with weights θ and $Q(s, a; \theta) \approx Q^*(m, a)$. The loss function to train the Q-network is shown in Eq. (2):

$$L_i(\theta_i) = E_{s,a \sim \rho(.)}[(y_i - Q(s, a; \theta_i))^2]. \tag{2}$$

The y_i is the target, which is calculated by the previous iteration result θ_{i-1}. The $\rho(m, a)$ is the probability distribution of sequences m and a. The gradient of the loss function is shown in Eq. (3):

$$\nabla_{\theta_i} L_i(\theta_i) = E_{m,a \sim \rho(.); s' \sim \xi}[(y_i - Q(m, a; \theta_i)) \nabla_{\theta_i} Q(m, a; \theta_i)]. \tag{3}$$

We consider tasks in which an agent interacts with an environment. In this case, the AAV moves step by step in a sequence of observations, actions and rewards. We took in consideration AAV mobility and consider 7 mobile patterns (`up`, `down`, `forward`, `back`, `left`, `right`, `stop`). In order to decide the reward function, we considered Distance between AAV and Obstacle (DAO) parameter.

The initial weights values are assigned as Normal Initialization [26]. The input layer is using AAV and the position of destination, total reward values in Experience Memory and AAV movements patterns. The hidden layer is connected with 256 rectifier units in Rectified Linear Units (ReLU) [27]. The output Q-values are the AAV movement patterns.

Algorithm 1. Tabu List for TLS-DQN.

Require: The coordinate with the highest evaluated value in the section is (x, y, z).

if $(x_{before} \leq x_{current}) \wedge (x_{current} \leq x)$ **then**
  $tabu\ list \Leftarrow ((x_{min} \leq x_{before}) \wedge (y_{min} \leq y_{max}) \wedge (z_{min} \leq z_{max}))$
else if $(x_{before} \geq x_{current}) \wedge (x_{current} \geq x)$ **then**
  $tabu\ list \Leftarrow ((x_{before} \leq x_{max}) \wedge (y_{min} \leq y_{max}) \wedge (z_{min} \leq z_{max}))$
else if $(y_{before} \leq y_{current}) \wedge (y_{current} \leq y)$ **then**
  $tabu\ list \Leftarrow ((x_{min} \leq x_{max}) \wedge (y_{min} \leq y_{before}) \wedge (z_{min} \leq z_{max}))$
else if $(y_{before} \geq y_{current}) \wedge (y_{current} \geq y)$ **then**
  $tabu\ list \Leftarrow ((x_{min} \leq x_{max}) \wedge (y_{before} \leq y_{max}) \wedge (z_{min} \leq z_{max}))$
else if $(z_{before} \leq z_{current}) \wedge (z_{current} \leq z)$ **then**
  $tabu\ list \Leftarrow ((x_{min} \leq x_{max}) \wedge (y_{min} \leq y_{max}) \wedge (z_{min} \leq z_{before}))$
else if $(z_{before} \geq z_{current}) \wedge (z_{current} \geq z)$ **then**
  $tabu\ list \Leftarrow ((x_{min} \leq x_{max}) \wedge (y_{min} \leq y_{max}) \wedge (z_{before} \leq z_{max}))$

3 Proposed Method

3.1 TLS-DQN

The idea of the Tabu List Strategy (TLS) is motivated from Tabu Search (TS) proposed by F. Glover [28] to achieve an efficient search for various optimization problems by prohibiting movements to previously visited search area in order to prevent getting stuck in local optima.

$$r = \begin{cases} 3 & (if\ (x_{current} = x_{global\ destinations}) \wedge \\ & (y_{current} = y_{global\ destinations}) \wedge \\ & (z_{current} = z_{global\ destinations})) \vee \\ & (((x_{before} < x_{current}) \wedge (x_{current} \leq x_{local\ destinations})) \vee \\ & ((x_{before} > x_{current}) \wedge (x_{current} \geq x_{local\ destinations})) \vee \\ & ((y_{before} < y_{current}) \wedge (y_{current} \leq y_{local\ destinations})) \vee \\ & ((y_{before} > y_{current}) \wedge (y_{current} \geq y_{local\ destinations})) \vee \\ & ((z_{before} < z_{current}) \wedge (z_{current} \leq z_{local\ destinations})) \vee \\ & ((z_{before} > z_{current}) \wedge (z_{current} \geq z_{local\ destinations}))). \\ -1 & (else). \end{cases} \tag{4}$$

In this paper, reward value is decided by Eq. (4), where "x", "y" and "z" means X-axis, Y-axis and Z-axis, respectively. The `current` means the current coordinates of the actor node in the DQN, and the `before` means the coordinates before selecting and moving the action. Also, the `global destination` means the destination in the problem area, and the `local destination` means the target passage points until the global destination.

The considered area is partitioned based on the target passage points and one destination is set in each area. If the current coordinate is closer to the destination than the coordinate before the move, also if the current coordinate is equal to the destination, the reward value is 3. In all other cases, the reward value is -1. The tabu list in TLS is used when an actor node of DQN selects

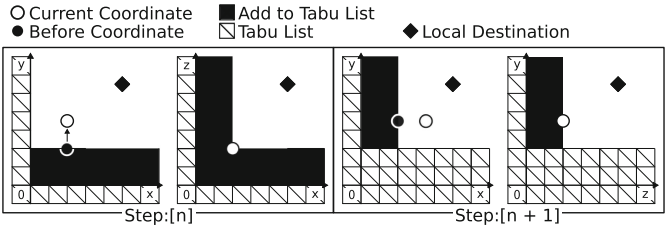


Fig. 4. Tabu rule addition method.

an action or the reward for that action is determined. The tabu list is referred to when selecting the action if the direction of movement of the action has been randomly determined. If the direction of movement area is included in the tabu list, the actor node will reselect the action. Also, the tabu list is used when the reward was determined. When the reward value is 3, the prohibited area is added to the tabu list based on the rule shown in Algorithm 1. The tabu list holds the added prohibited areas until the end of the episode and is initialized for each episode.

Figure 4 shows an example of adding the prohibited area method to the tabu list according to Algorithm 1. The n in Fig. 4 is a natural number and refers to the number of iterations in each episode. In Step: $[n]$ of Fig. 4, the actor node has moved in the Y-axis direction and is closer to the destination than before the move, and $(y_{before} < y_{current})$ and $(y_{current} \leq y_{local\ destinations})$ in Algorithm 1 are satisfied. Therefore, the black-filled area of $[(x_{min} \leq x_{max}), (y_{min} \leq y_{before}), (z_{min} \leq z_{max})]$ is added to the tabu list. Also, in Step: $[n+1]$, the actor node has moved in the X-axis direction and is closer to the destination than before the move, and $(x_{before} < x_{current})$ and $(x_{current} \leq x_{local\ destinations})$ in Algorithm 1 are satisfied. Therefore, the black-filled area with $[(x_{min} \leq x_{before}), (y_{min} \leq y_{max}), (z_{min} \leq z_{max})]$ is added to the tabu list.

The search by TLS-DQN is done in a wider range and is better than the search by random direction of movement.

3.2 LiDAR Based Mobile Area Decision Method

The proposed method reduces the settings operation for the destination and the division of mobile area, which was set manually by humans in TLS-DQN. In addition, the proposed method can decide the destination with less computation than using SLAM or other methods. In Algorithm 2, using as inputs the coordinates list of obstacles (*distance*, *angle*) obtained by LiDAR and the coordinates of LiDAR placement is generated as output the *Destination* (X, Y) as `local destination` or `global destination`. Also, the Z-coordinate of destination is the median of the movable range in the Z-axis for `local destination` and the minimum of the movable range for `global destination`. The mobile area is divided based on the decided *Destination*. In TLS-DQN, if the actor node reached a local destination, the movable range is decided again based on LiDAR,

Algorithm 2. LiDAR Based Mobile Area Decision Method.

Input: *Point Cloud List* ← The coordinates list of obstacles (*distance*, *angle*) obtained by LiDAR
$(x_{LiDAR},\ y_{LiDAR})$ ← The coordinates of LiDAR Placement.
Output: *Destination* (X, Y).
for $i = 0$ **to** 360 **do**
  $x_{Point\ Cloud\ List}[i] \leftarrow Point\ Cloud\ List[i][0] \times cos(Point\ Cloud\ List[i][1])$.
  $y_{Point\ Cloud\ List}[i] \leftarrow Point\ Cloud\ List[i][0] \times sin(Point\ Cloud\ List[i][1])$.
  if $Point\ Cloud\ List[i][0] > Any\ Distance$ **then**
    $Distant\ Point\ Cloud[i] \leftarrow (x_{Point\ Cloud\ List}[i],\ y_{Point\ Cloud\ List}[i])$.
$(x_{\min},\ x_{\max})$ ← Min. and Max. value for X-axis in the *Distant Point Cloud*.
$(y_{\min},\ y_{\max})$ ← Min. and Max. value for Y-axis in the *Distant Point Cloud*.
$(x_{center},\ y_{center}) \leftarrow (\frac{x_{\min}+\ x_{\max}}{2},\ \frac{y_{\min}+\ y_{\max}}{2})$.
$flag \leftarrow 0$.
for $x = x_{LiDAR}$ **to** x_{center} **do**
  $y \leftarrow (\frac{y_{center}-\ y_{LiDAR}}{x_{center}-\ x_{LiDAR}}) \times (x - x_{LiDAR}) + y_{LiDAR}$.
  for $i = 0$ **to** 360 **do**
    if $\sqrt{(x - x_{Point\ Cloud\ List}[i])^2 + (y - y_{Point\ Cloud\ List}[i])^2} > Any\ Distance$ **then**
      $Destination \leftarrow (x,\ y)$.
    else
      $flag \leftarrow 1$.
      break

if $flag = 0$ **then**
  Destination is `local destination`.
else
  Destination is `global destination`.

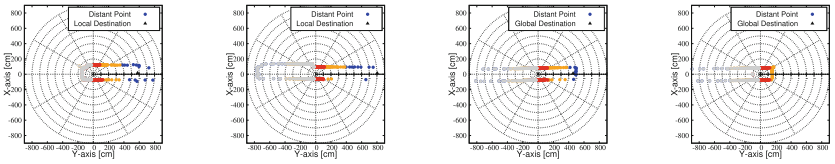
then the destination is updated. When updating, the destination is decided continuously by using the coordinates of the reached `local destination` for LiDAR placement.

(a) From the initial placement to the grobal destination. (b) From the grobal destination to the initial placement.

Fig. 5. Snapshot of considered area.

4 Performance Evaluation

In this section, we describe the experimental results of the LiDAR based mobile area decision method for TLS-DQN to improve the control for AAV Mobility and the simulation results of TLS-DQN based on the decided area.



(a) Initial placement. (b) Local destination. (c) Local destination. (d) Global destination.

Fig. 6. Experimental results of the proposed method.

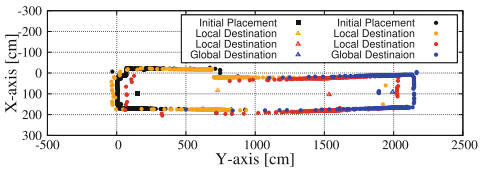


Fig. 7. Visualization results of stitching the considered area.

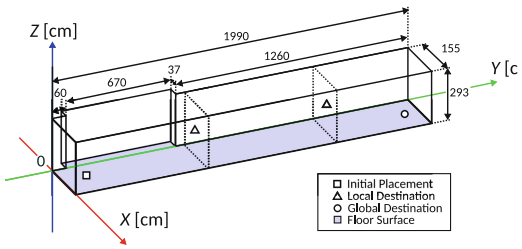


Fig. 8. Considered area for simulation.

4.1 Results of LiDAR Based Mobile Area Decision Method

The target environment is a corridor with an indoor single-path environment. Figure 5 shows snapshots of the area used in the simulation scenario and it was

taken on the ground floor of Building C4 at Okayama University of Science, Japan. In Fig. 6 are shown the experimental results of the proposed method for initial placement, Local Destination and Global Destination. In Fig. 6, red points indicate short distance, orange points indicate medium distance, blue points indicate distant distance and gray points indicate points behind the direction of movement. While, Fig. 7 shows the visualization results when switching the considered area. In Fig. 6 and Fig. 7, the red points indicate short distance, orange points indicate medium distance, blue points indicate long distance and gray points indicate points behind the direction of movement. Figure 8 shows the considered area based on the actual measurements of Fig. 5 area. By using the proposed method, for the initial configuration [100, 125, 0], the destinations are [90, 700, 150], [100, 1550, 150], [90, 1925, 0] (`global destination`) in this scenario. The experimental results show that the `global destination` is decided between the initial placement to the end of the path, and the `global destination` is decided at the end of the path.

Table 2. Simulation parameters of DQN.

Parameters	Values
Number of episode	10000
Number of iteration	2000
Number of hidden layers	3
Number of hidden units	15
Initial weight value	Normal initialization
Activation function	ReLU
Action selection probability (ϵ)	$0.999 - (t$ / Number of episode) ($t = 0, 1, 2, \ldots,$ Number of episode)
Learning rate (α)	0.04
Discount rate (γ)	0.9
Experience memory size	300×100
Batch size	32
Number of AAV	1

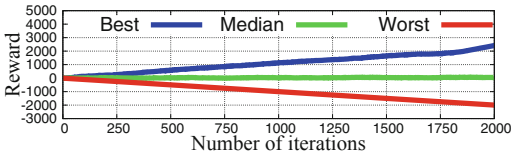


Fig. 9. Simulation results of rewards.

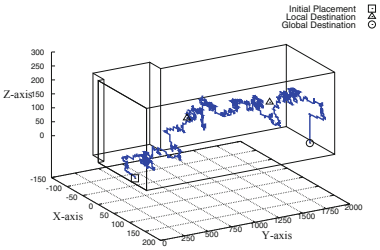


Fig. 10. Visualization results.

4.2 Simulation Results of TLS-DQN

We consider for simulations the operations such as takeoffs, flights and landings between the initial position and the destination. Figure 8 shows the considered area for simulation. Table 2 shows the parameters used for simulations. The `global destination` and `local destination` is decided by the proposed method. Figure 9 shows the change in reward value of the action in each iteration for `Worst`, `Median`, and `Best` episodes in TLS-DQN. In the episode of `Best`, it can be seen that the reward value is on the rise trend. Figure 10 shows the visualization results of Best episodes in TLS-DQN. The visualization results show that TLS-DQN is reaching the global destination.

5 Conclusions

In this paper, we proposed a LiDAR based mobile area decision method for TLS-DQN to improve the control for AAV mobility. The proposed method is used in an indoor single-path environment and the simulations were carried out by TLS-DQN for AAV control. From the evaluation results, we conclude as follows.

- The proposed method can decide the mobile area and destination based on LiDAR.
- The proposed method has a less computation than other methods.
- The proposed method is a good approach for indoor single-path environments.

In the future, we would like to improve the TLS-DQN for AAV mobility and will consider different scenarios.

Acknowledgement. This work was supported by JSPS KAKENHI Grant Number JP20K19793 and Grant for Promotion of OUS Research Project (OUS-RP-20-3).

References

1. Stöcker, C., et al.: Review of the current state of UAV regulations. Remote Sens **9**(5), 1–26 (2017)
2. Artemenko, O., et al.: Energy-aware Trajectory Planning for the Localization of Mobile Devices using an Unmanned Aerial Vehicle. In: Proceedings of The 25-th International Conference on Computer Communication and Networks (ICCCN-2016), pp. 1–9 (2016)
3. Popović, M., et al.: An informative path planning framework for UAV-based terrain monitoring. Auton. Robots **44**, 889–911 (2020)
4. Nguyen, H., et al.: LAVAPilot: Lightweight UAV Trajectory Planner with Situational Awareness for Embedded Autonomy to Track and Locate Radio-tags. arXiv:2007.15860, pp. 1–8 (2020)
5. Oda, T., et al.: Design and implementation of a simulation system based on deep q-network for mobile actor node control in wireless sensor and actor networks. In: Proceedings of of The 31-th IEEE International Conference on Advanced Information Networking and Applications Workshops (IEEE AINA-2017), pp. 195–200 (2017)
6. Oda, T., et al.: Performance evaluation of a deep q-network based simulation system for actor node mobility control in wireless sensor and actor networks considering three-dimensional environment. In: Proceedings of The 9-th International Conference on Intelligent Networking and Collaborative Systems (INCoS-2017), pp. 41–52 (2017)
7. Oda, T., Kulla, E., Katayama, K., Ikeda, M., Barolli, L.: A deep q-network based simulation system for actor node mobility control in WSANs considering three-dimensional environment: a comparison study for normal and uniform distributions. In: Barolli, L., Javaid, N., Ikeda, M., Takizawa, M. (eds.) CISIS 2018. AISC, vol. 772, pp. 842–852. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-93659-8_77
8. Saito, N., Oda, T., Hirata, A., Hirota, Y., Hirota, M., Katayama, K.: Design and implementation of a DQN based AAV. In: Barolli, L., Takizawa, M., Enokido, T., Chen, H.-C., Matsuo, K. (eds.) BWCCA 2020. LNNS, vol. 159, pp. 321–329. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-61108-8_32
9. Sandino, J., et al.: UAV framework for autonomous onboard navigation and people/Object detection in cluttered indoor environments. Remote Sens. **12**(20), 1–31 (2020)
10. Scherer, J., et al.: An autonomous Multi-UAV system for search and rescue. In: Proceedings of The 6-th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (DroNet-2015), pp. 33–38 (2015)
11. Moulton, J., et al.: An Autonomous Surface Vehicle for Long Term Operations. In: Proceedings of MTS/IEEE OCEANS, pp. 1–10 (2018)
12. Oda, T., et al.: Design of a Deep Q-Network Based Simulation System for Actuation Decision in Ambient Intelligence. In: Proceedings of The 33-rd International Conference on Advanced Information Networking and Applications (AINA-2019), pp. 362–370 (2019)
13. Oda, T., et al.: Design and implementation of an IoT-based E-learning testbed. Int. J. Web Grid Serv. **13**(2), 228–241 (2017)

14. Hirota, Y., Oda, T., Saito, N., Hirata, A., Hirota, M., Katatama, K.: Proposal and experimental results of an ambient intelligence for training on soldering iron holding. In: Barolli, L., Takizawa, M., Enokido, T., Chen, H.-C., Matsuo, K. (eds.) BWCCA 2020. LNNS, vol. 159, pp. 444–453. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-61108-8_44
15. Hayosh, D., et al.: Woody: low-cost, open-source humanoid torso robot. In: Proceedings of The 17-th International Conference on Ubiquitous Robots (ICUR-2020), pp. 247–252 (2020)
16. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015)
17. Mnih, V., et al.: Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602, pp. 1–9 (2013)
18. Lei, T., Ming, L.: A robot exploration strategy based on q-learning network. In: IEEE International Conference on Real-time Computing and Robotics (IEEE RCAR-2016), pp. 57–62 (2016)
19. Riedmiller, M.: Neural fitted Q iteration- First experiences with a data efficient neural reinforcement learning method. In: Proceedings of The 16-th European Conference on Machine Learning (ECML-2005), pp. 317–328 (2005)
20. Lin, L.J.: Reinforcement Learning for Robots Using Neural Networks. In: Proceedings of Technical Report, DTIC Document (1993)
21. Lange, S., Riedmiller, M.: Deep auto-encoder neural networks in reinforcement learning. In: Proceedings of The International Joint Conference on Neural Networks (IJCNN-2010), pp. 1–8 (2010)
22. Kaelbling, L.P., et al.: Planning and acting in partially observable stochastic domains. Artif. Intell. **101**(1–2), 99–134 (1998)
23. Saito, N., Oda, T., Hirata, A., Nagai, Y., Hirota, M., Katayama, K.: Proposal and evaluation of a tabu list based DQN for AAV mobility. In: Barolli, L., Natwichai, J., Enokido, T. (eds.) EIDWT 2021. LNDECT, vol. 65, pp. 189–200. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-70639-5_18
24. Saito, N., et al.: A Tabu list strategy based DQN for AAV mobility in indoor single-path environment: Implementation and performance evaluation. Internet of Things **14**, 100394 (2021)
25. Takano, K., et al.: Design of a DSL for converting rust programming language into RTL. In: Proceedings of The 8-th International Conference on Emerging Internet, Data & Web Technologies (EIDWT-2020), pp. 342–350 (2020)
26. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of The 13-th International Conference on Artificial Intelligence and Statistics (AISTATS-2010), pp. 249–256 (2010)
27. Glorot, X., et al.: Deep sparse rectifier neural networks. In: Proceedings of The 14-th International Conference on Artificial Intelligence and Statistics (AISTATS-2011), pp. 315–323 (2011)
28. Glover, F.: Tabu search - part I. ORSA J. Comput. **1**(3), 190–206 (1989)