Drone path planning method to reduce energy consumption

Takuto Maejima[1*] , Xiangbo Kong[2†] , Tomoyasu Shimada[1†] , Hiroki Nishikawa[3†], and Hiroyuki Tomiyama[1†]

[1]*Graduate School of Science and Engineering, Ritsumeikan University*
[2]*Department of Intelligent Robotics, Factory of Engineering, Toyama Prefectural University*
[3]*Graduate School of Information Science and Technology, Osaka University*

Abstract

In this study, two methods are proposed to reduce the energy consumption of drones. The first method is path planning using depth images, which combines the path planning algorithm of the previous study with the algorithm for speed control in the direction of travel. The second method inputs depth images into deep reinforcement learning to learn speed control and collision avoidance. This method also uses the algorithms of previous studies for path planning. As it is difficult to fly drones freely and conduct experiments due to the severe punishment of drone flight laws in recent years, experiments are conducted using a drone simulator called AirSim. Also, since energy consumption cannot be directly obtained on AirSim, we propose a method for calculating energy consumption on AirSim. The experimental results show that the time required for the arrival of the destination was shortened and the energy consumption was drastically reduced in both proposed methods compared with the previous study. Specifically, the path planning method using speed control in the traveling direction reduced the arrival time by about 39 seconds and the energy consumption by about 42% compared with the previous study. Also, the path planning method using deep reinforcement learning reduced the arrival time by about 47 seconds and the energy consumption by about 30% compared with the previous study. In addition, the collision rate was improved by about 2.8% in the path planning method using speed control in the traveling direction and by about 3.8% in the path planning method using deep reinforcement learning compared with the previous study.

Keywords

drone, depth image, path planning, AirSim, deep reinforcement learning, energy consumption

1. Introduction

In recent years, drones have attracted strong interest all over the world, and the market size is growing year by year. Drones are expected to be used in a variety of scenes by developments (e.g. package delivery, survey of disaster-affected areas, aerial photography). However, drones have some problems, such as not being able to carry heavy loads due to weight restrictions, not flying on optimal paths, and not colliding with each other [1][2]. In addition, most of the current mainstream drones are battery-powered, which means they need to fly with limited power. Moreover, in recent years, it has become difficult to fly drones freely and conduct experiments due to the severe punishment of drone flight laws [3].

Therefore, this study was conducted for the following two purposes. The first is the use of drone simulators to conduct experiments freely since it is difficult to conduct sufficient experiments in the real world due to strict legal regulations. In the experiment, a simulator named AirSim was used. The second is to fly to the destination with a low collision rate and low energy consumption. In the

ATAIT 2023: The 5th International Symposium on Advanced Technologies and Applications in the Internet of Things, August 28–29, 2023, Kusatsu, Shiga, Japan
*Corresponding author.
†These authors contributed equally.
EMAIL: ri0106er@ed.ritsumei.ac.jp (T. Maejima); kong@pu-toyama.ac.jp (X. Kong); tomoyasu.shimada@tomiyama-lab.org (T. Shimada); nishikawa.hiroki@ist.osaka-u.ac.jp (H. Nishikawa); ht@fc.ritsumei.ac.jp (H. Tomiyama)

experiment, to reduce the time it takes to reach the destination and to reduce energy consumption, path planning using speed control in the traveling direction of the drone is carried out. In addition, we thought that collision avoidance with minimal action would also reduce energy consumption, so we used deep reinforcement learning to control the speed in the traveling direction and avoid collisions.

The structure of this paper is as follows. Section 2 describes the relevant research of this study. Section 3 describes an algorithm for speed control in the traveling direction, which we devised to solve the problems of the previous study, and an algorithm for computing energy consumption on AirSim. In Section 4, the environment and experimental results of deep reinforcement learning and learning of speed control and collision avoidance using depth images are described. In Section 5, the summary of this paper and future issues are described.

2. Related study

This section describes the relevant research of this study. In paper [4], LiDAR was used to study obstacle detection and collision avoidance of drones. Although LiDAR has high performance, it is not suitable for installation on small drones due to its high cost and weight. In paper [5], a depth image acquired from a drone is divided into 289 sections to determine the safest direction from each of the sections to avoid a drone collision. However, in paper [5], experiments are conducted using actual drones, but the experimental environment is simple and not suitable for conducting experiments on collision avoidance, which is a problem that the experimental environment is insufficient. Study [6] proposes a path planning method for drones by improving the collision avoidance method in paper [5]. In paper [6], they used a drone simulator named AirSim for experiments and a map suitable for experiments distributed for simulation. Several literature studies are showing the effectiveness of AirSim, and in paper [7], they conducted a comparative experiment between flight in real space and flight on AirSim and showed that the flight characteristics of a drone on AirSim are close to those in real space. Also, in paper [8], it was shown that AirSim is suitable for experiments such as deep learning because it can acquire data and images without delay. Therefore, in paper [6], it can be said that the problem of experimenting in an insufficient environment in paper [5] has been solved. However, in paper [6], there is a problem in that the speed of the drone in the traveling direction is slow and constant, resulting in poor power efficiency and large energy consumption required to reach the destination.

In addition, in recent years, reinforcement learning and deep reinforcement learning have been increasingly used in drone research to improve the performance of autonomous flight. In paper [9], AirSim and reinforcement learning were used to learn how to land a drone safely, and the learned model was transferred to a real vehicle to accomplish the task in a real vehicle with little learning on the simulator. It was shown that the cost and time required for real machine learning could be drastically reduced by advanced learning on the simulator. However, study [9] has a problem that is not realistic because the learning content of reinforcement learning of drones is only landing and does not carry out complicated tasks such as collision avoidance. In paper [10], they used deep reinforcement learning to learn how to plan a path to a destination while avoiding collisions on an AirSim. However, problems in paper [10] are that the environment for experiments is insufficient because the destination is set in a straight line of the drone's camera and the number of obstacles between the initial point and the destination is small. The approach of [11] proposes a path planning algorithm that performs collision avoidance while saving power based on deep reinforcement learning. In the experiment, they can say that the problems of paper [10] are solved because the simulator is used in an environment where there are many obstacles and the destination is not in a straight line. Furthermore, the learned model achieved the task not only in the learned environment but also in the unknown environment without significantly reducing the collision rate. This shows that in drone collision avoidance, the deep reinforcement learning-based algorithm can accomplish the task regardless of the environment. However, there is a problem in paper [11] that learning is performed by setting the movement speed of drones at a constant speed. However, papers [10], [11] and [12] have a problem in that they do not take advantage of the vertical movement that is a characteristic feature of drones because the flight altitude is fixed.

To address these issues, we propose a path planning method for drones with low energy consumption in this paper.

3. Path planning method using direction-of-travel speed control

This section describes a path planning method using speed control in the direction of a drone's travel, which aims to reduce power consumption by reducing the time it takes to reach a destination.

3.1. Previous study

In this section, we describe collision avoidance and path planning algorithms using depth images in paper [6]. Collision avoidance is divided into two algorithms: a direction-specifying algorithm that determines the direction a drone flies from depth images and an algorithm that performs speed control during flight.

First, we describe a direction-specifying algorithm. we acquire a depth image of 256 × 144 pixels from AirSim. Then, we divide the depth image into 289 sections, each 17 by 17 in length and width, as in paper [5]. Each section is 69 × 42 pixels in size, and the sections overlap by shifting 11 pixels to the right and 6 pixels down. Also, each section is assigned a number and managed in a coordinate format such that the top left section is (0, 0), the center section is (8, 8), and the bottom right section is (16, 16). After dividing the sections, the pixel values of each section are calculated, and if the average of the pixel values of the center section is greater than the set threshold, the center section is set as the largest section, otherwise, the section with the largest pixel value is selected from the sections other than the center, and that section is set as the largest section. Finally, collision avoidance is realized if the speed is controlled so that the selected maximum section becomes the central section in the next process. Then, an algorithm for speed control is described. In the algorithm for speed control, speed control is performed using PID control so that the maximum section selected in the algorithm for collision avoidance moves to the position of the center section.

Finally, an algorithm for path planning to a destination is described. First, the rotation angle of the drone is calculated from the coordinates of the current drone and the coordinates of the destination. Then, when the drone is not performing collision avoidance, that is, when the input speed on the Y and Z axes is 0 when the direction in which the drone's camera is facing is taken as the X axis, and when the calculated rotation angle is larger than the threshold defined as the allowable angle deviation, the drone can be redirected toward the destination by rotating the drone by that rotation angle.

In this study, to avoid a collision between a drone and an object as much as possible, the speed of the drone in the direction of travel is fixed at a low speed, and since the speed remains low even in a situation where collision avoidance is not performed, the time required to arrive at the destination is long and the energy consumption is large. Therefore, we propose a method to improve energy consumption, which is a problem, without lowering the collision rate by referring to the previous study.

3.2. Proposed method

In this section, we describe an algorithm for speed control in the traveling direction, which was inspired by a paper [6], and an algorithm for calculating power consumption on AirSim.

3.2.1. Algorithm for speed control in the direction of travel

In this section, we describe an algorithm for controlling the speed of a drone in its traveling direction. In the AirSim environment, the coordinate axes relative to the drone are shown in Fig. 3.1, and in this study, the velocity of the drone in the X-axis direction is vx, the velocity in the Y-axis direction is vy, and the velocity in the Z-axis direction is vz. Therefore, the velocity control in the traveling direction in this experiment is to control vx.

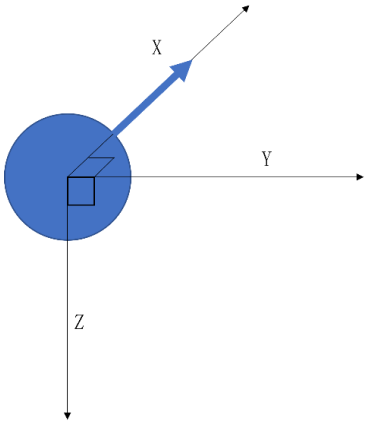

Fig. 3.1 Coordinate axis of the drone in AirSim [7]

First, when the uppermost left coordinate of the acquired depth image is (x, y) = (0, 0), a 21 × 69 rectangular section such as Fig. 3.2 with (x, y) = (88, 48) as the upper left corner is cut out to make vx _ section.

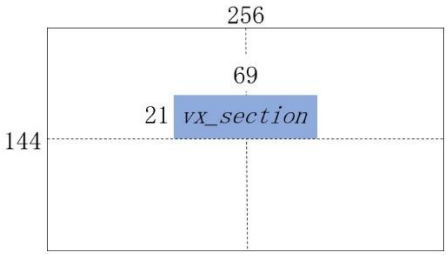

Fig. 3.2 vx _ section

Next, we prepare a variable avg _ vx _ section that serves as a threshold for controlling vx and store the average of the pixel values in vx _ section in avg _ vx _ section = 50 if the drone is moving to avoid obstacles, or in avg _ vx _ section if it is not. Then, by preparing limit _ vx, which is a variable for setting the upper limit of vx, limit _ vx = 9.0 if the value of avg _ vx _ section is 200 or more, 6.0 if it is 150 or more but less than 200, and 3.0 otherwise. Finally, it adds 0.01 to vx when vx is less than limit _ vx and subtracts 0.01 when vx is greater than limit _ vx. The above flow can be expressed as Eq. (3.1) and Eq. (3.2).

$$limit\_vx = \begin{cases} 9.0\ (avg\_vx\_section \geq 200) \\ 6.0\ (200 > avg\_vx\_section \geq 150) \\ 3.0\ (150 > avg\_vx\_section) \end{cases} \tag{3.1}$$

$$vx = \begin{cases} vx + 0.01(vx < limit\_vx) \\ vx - 0.01(vx > limit\_vx) \end{cases} \tag{3.2}$$

This reduces the large inclination of the drone due to a sudden change in speed while controlling the drone to be fast when there are no obstacles in the direction of travel and slow when there are obstacles.

3.2.2. Algorithm for energy consumption calculation

In this section, we describe an algorithm for calculating the energy consumption of drones. In calculating the power consumption of a drone, we divide the calculation into two parts: the power consumed by the motor that turns the propeller required for the drone to fly, and the power consumed other than the motor.

First, we explain how to calculate power consumption in the motor. Since AirSim cannot obtain the values of the motor voltage and the current flowing there, which are necessary for calculating power consumption, we obtain power consumption from motor power. The motor power P' [W] can be expressed by Eq. (3.3), where the motor angular velocity is denoted by ω [rad/s], and the motor torque is denoted by T [N · m].

$$P' = \omega T \tag{3.3}$$

The output per unit time of a motor is called power, and to obtain this power, electrical power must be supplied to the motor input. However, not all of the supplied power is converted into output power, as some of the power is lost as heat energy and other losses. The ratio of the supplied power to the output power is known as motor efficiency. Considering the motor efficiency as n, the power consumption taking into account the motor efficiency can be calculated as P_motor [W], and the power consumption is obtained by Eq. (3.4).

$$P\_motor = P'\frac{1}{n} \tag{3.4}$$

Next, the calculation method of power consumption other than the motor is explained. Drones also have cameras that capture depth images [13] and small boards [14] in addition to motors. In experiments, the power consumption of these devices was determined as a single constant, and the power consumption was calculated by multiplying the flight time of the drone by this constant. Power consumption per unit time other than the motor is expressed as q [W] and the drone flight time is expressed as t [s] and the total drone energy consumption in one flight is expressed as P [Ws], energy consumption is obtained by Eq. (3.5).

$$P = \left(\omega T\frac{1}{n} + q\right)t \tag{3.5}$$

3.3. Comparative experiment

This section describes the contents, experimental results, and discussion of the comparison with the previous study [6]. In this experiment, we compare the average energy consumption, the collision rate, and the average arrival time at the destination for each method at 500 randomly selected points in the range of $100 \leqq |x|, |y| \leqq 200$. Also, we use a distribution map called Blocks [15] where many static obstacles of various shapes are placed as shown in Fig. 3.3. The motor efficiency is n = 0.8 and the Power consumption per unit time other than the motor is q = 10.5. In this experiment, power consumption was determined based on the specifications of the NVIDIA Jetson Orin Nano [14] as the single-board computer and the Intel RealSense™ Depth Camera D400 series [13] as the depth sensor camera. In this method, the power consumption is set to the minimum specified in the documentation (7W) plus the power consumption of the depth sensor camera (3.5W).

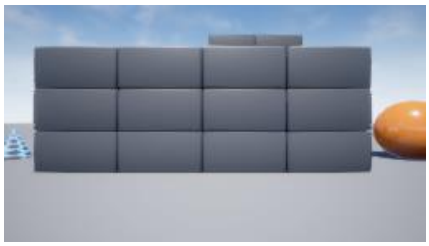

Fig. 3.3 Blocks [15]

The experimental environment is as follows.

- CPU: Intel Core i7-9750H
- GPU: NVIDIA GeForce GTX 1650
- RAM: 16GB
- AirSim v1.8.1
- Unreal Engine 4.25.4

The experimental results are shown in Table 3.1.

Table 3.1 Experimental results

	Average energy consumption [Ws]	Collision rate [%]	Average arrival time [s]
The previous study [6]	7483.8	7.8	89.5
Proposed method	4315.0	5.0	50.8

Table 3.1 shows that the average arrival time of the proposed method with speed control in the traveling direction was shortened by about 39 seconds and the average energy consumption was reduced by about 42% compared with the previous study method. Therefore, in an environment with only static obstacles such as Blocks, it was found that if the arrival time to the destination was shortened by increasing the speed, power consumption per unit time would be larger due to the increase in the motor speed, but the overall energy consumption would be smaller.

Furthermore, Table 3.1 shows that the collision rate of the proposed method is about 2.8% lower than that of the previous study. This is because when the vx of the drone enters the collision avoidance state at a speed faster than 3 m/s, negative acceleration is applied to the vx and the speed is reduced to 3 m/s. Since the drone's attitude tilts slightly due to the acceleration, the drone's direction of travel is slightly more diagonally upward than normal when the acceleration is negative. As a result, it is thought that the velocity of vx affects the velocity in the upward direction, making it easier for the vehicle to pass over obstacles in a large way, thus lowering the collision rate.

4. Drone speed control and collision avoidance using deep reinforcement learning

This section describes drone speed control and collision avoidance using depth images and deep reinforcement learning. The proposed method described in Section 3 aims to achieve more optimal speed control and collision avoidance by using deep reinforcement learning because the speed is constant at low speed during collision avoidance. Also, the path planning to the destination of the drone is based on the route planning algorithm proposed in the previous study [6].

4.1. Learning environment

This section describes the environment, agents, rewards, and hyperparameters of deep reinforcement learning set up in the experiment.

4.1.1. Environment and Agent

First, the environment is described. The algorithm of deep reinforcement learning uses Proximal Policy Optimization (PPO) [16] of Stable Baselines 3[17]. The policy network uses CNN because depth images are used for input. For learning, a map called Blocks [15] is used, where the initial point of the drone is (x, y, z) = (0,0, -9), and the initial velocity is set to (vx, vy, vz) = (0, 0, 0), where vx is the velocity in the X-axis of the drone, vy is the velocity in the Y-axis, and vz is the velocity in the Z-axis.

Next, the agent is described. The agent is a drone that can be operated by the API provided by AirSim, and the following seven actions can be taken by the agent.

- $vx = vx + 0.25,\ vy = vy,\ vz = vz$
- $vx = vx - 0.25,\ vy = vy,\ vz = vz$
- $vx = vx,\ vy = vy + 0.25,\ vz = vz$
- $vx = vx,\ vy = vy - 0.25,\ vz = vz$
- $vx = vx,\ vy = vy,\ vz = vz + 0.25$
- $vx = vx,\ vy = vy,\ vz = vz - 0.25$
- $vx = vx,\ vy = vy,\ vz = vz$

4.1.2. Reward

In this section, we describe the rewards set in learning. The ideal control of drones that we want to realize by reinforcement learning is a movement that increases the traveling speed as much as possible while not colliding with obstacles.

Next, the reward set in the experiment is explained. The reward consists of two parts: reward _ speed and reward _ state. The reward _ speed is a reward for the forward velocity vx of the drone. When the vx of the current drone is between 3 ~ 10 m/s, the reward is vx multiplied by 0.05, and when vx is 0 ~ 3 m/s, the reward is -0.05, otherwise, the reward is -0.1. Therefore, when the speed of vx is fast, the reward is given positively, and when the speed is slower than a certain value, the reward is given negatively. The reward _ speed is expressed as Eq. (4.1).

$$reward\_speed = \begin{cases} 0.05vx\ (3 \leq vx \leq 10) \\ -0.05\ (0 \leq vx \leq 3) \\ -0.1 \end{cases} \tag{4.1}$$

Next, the reward _ state is explained. The purpose of this reward is to keep the motion during collision avoidance as small as possible by keeping the non-traveling velocities vy and vz as zero as possible during the flight. If vx of the traveling speed is greater than 0 and the other speeds vy and vz are 0, the reward is 0.05, otherwise 0. The reward _ state is expressed as Eq. (4.2).

$$reward\_state = \begin{cases} 0.05\ (vx > 0 \cap vy == 0 \cap vz == 0) \\ 0 \end{cases} \tag{4.2}$$

The total reward _ speed and reward _ state are given as a reward, but if a drone strikes, the reward is -10, and done = 1, which is the judgment for ending an episode. Therefore, if the reward is expressed as a reward, the reward looks like equation (4.3).

$$reward = \begin{cases} reward\_speed + reward\_state\ (\text{If the drone has not collided}) \\ -10\ (\text{In case of drone collisions}) \end{cases} \tag{4.3}$$

Next, Other exceptional rewards and termination conditions are explained. If there is a drone within a 1-meter radius of the destination, it is determined that the drone has reached the destination and done = 1. This is because, if the arrival judgment is made only in the coordinates of the destination, it is necessary to pass exactly in the coordinates of the destination for the timing of the arrival judgment, but the judgment is severe for a drone with high speed, and reinforcement learning is also difficult to progress. In addition, if the drone somehow goes below the ground level, because the behavior is impossible in reality, reward = -10 is set and done = 1. In addition, if the drone is more than 400 m away from the initial point, if the current height of the drone exceeds 150 m, or if the flying time during one flight exceeds 1000 seconds, the learning efficiency is poor even if the drone is allowed to continue learning, so done = 1.

4.1.3. Hyperparameters

In this section, we list the hyperparameters. the parameters used in the experiment are shown in Table 4.1. The names of the parameters used in the experiment are based on Stable Baselines 3 [17].

Table 4.1 Hyperparameters used in the experiment

Parameter name	Set value
$gamma$	0.99
$learning\_rate$	0.00025
gae_lambda	0.95
n_steps	2048
$batch\_size$	512
n_epochs	4
$clip\_range$	0.2
$normalize\_advantage$	True
ent_coef	0.0005
vf_coef	0.5
max_grad_norm	0.5
$total\_timesteps$	500000

4.2. Comparative experiment

In this section, we compare the model learned by deep reinforcement learning with other methods. The comparison methods are the path planning method using speed control in the traveling direction proposed in Section 3 and the previous study [6] introduced in the 3.1 section.

In this experiment, we compare the average energy consumption, the collision rate, and the average arrival time at the destination for each method at 500 randomly selected points in the range of $100 \leqq |x|, |y| \leqq 200$. Also, we use Blocks [15]. We also set the motor efficiency at n = 0.8, q = 10.5 for power consumption other than the motor using the method without deep reinforcement learning, and q = 18.5 for power consumption other than the motor using the method with deep reinforcement learning. In this experiment, power consumption was determined based on the specifications of the NVIDIA Jetson Orin Nano [14] as the single-board computer and the Intel RealSense™ Depth Camera D400 series [13] as the depth sensor camera. The reason for setting the power consumption to 10.5W for the method that does not utilize deep reinforcement learning and 18.5W for the method that utilizes deep reinforcement learning is that it is believed that employing deep reinforcement learning increases the power consumption of the single-board computer installed in the drone. In the method that does not utilize deep reinforcement learning, the power consumption is set to the minimum specified in the documentation (7W) plus the power consumption of the depth sensor camera (3.5W). In the method that utilizes deep reinforcement learning, the power consumption is set to the maximum specified in the documentation (15W) plus the power consumption of the depth sensor camera (3.5W).

The experimental environment is as follows.

- CPU: Intel Core i7-9750H
- GPU: NVIDIA GeForce GTX 1650
- RAM: 16GB
- AirSim v1.8.1
- Unreal Engine v4.25.4
- Stable Baselines 3 v1.6.2
- OpenAI Gym v0.21.0 [18]

The experimental results are shown in Table 4.2.

Table 4.2 Experimental results

	Average energy consumption [Ws]	Collision rate [%]	Average arrival time [s]
The previous study [6]	7483.8	7.8	89.5
Proposed method (Direction-of-travel speed control)	4315.0	5.0	50.8
Proposed method (Deep reinforcement learning)	5301.3	4.0	42.4

From the experimental results in Table 4.2, the proposed method using deep reinforcement learning was able to improve all 3 items of energy consumption, collision rate, and average arrival time compared to the previous study. Specifically, the path planning method using speed control in the traveling direction reduced the average arrival time by about 39 seconds and the average energy consumption by about 42% compared with the previous study. The path planning method using deep reinforcement learning also reduced the average arrival time by about 47 seconds and the average energy consumption by about 30%. The collision rate was improved by about 2.8% in the path planning method using speed control in the traveling direction and by about 3.8% in the path planning method using deep reinforcement learning compared with the previous study.

The method using deep reinforcement learning results in a shorter average arrival time but larger energy consumption than the path planning method using speed control in the traveling direction. There are two possible reasons why the energy consumption of the method using deep reinforcement learning is larger than the path planning method using speed control in the traveling direction. The first reason, we believe, is that when deep reinforcement learning is used, power consumption per unit of time is large because more power consumption is put on the board than when it is not used. However, in an environment where the distance to the destination is longer than that of the present experiment or where there are many obstacles, the difference in the average arrival time between the method using deep reinforcement learning and the method using speed control in the traveling direction becomes larger, and the difference in power consumption of the board by using deep reinforcement learning is considered to have less impact.

The second reason, we believe, is that the lack of learning time has prevented us from minimizing collision avoidance, and even when there are no obstacles in front of us and we can go straight, the vy and vz values may not be zero. We believe this can be improved by further increasing the total number of learning steps. We also believe that by making the positive reward of the reward _ state larger than the current one, we can perform optimized path planning with minimal collision avoidance than the present model. However, if the reward in the reward _ state is increased, receiving a positive reward in the reward _ state will be prioritized over avoiding a collision, which may lead to a higher collision rate, so the balance with a negative reward in a collision must also be adjusted. In terms of collision rate, the proposed method using deep reinforcement learning showed the lowest rate compared to the previous study and the proposed method using speed control in the traveling direction. In this regard as well, we believe that the collision rate can be further reduced by increasing the total number of learning steps, which we have previously described as an improvement measure.

The improvement measures described so far are based on hyperparameters and rewards, but by giving not only depth images but also the current location and destination of energy consumption and drones as feature values to the input of deep reinforcement learning, the number of power consumption other than motors will increase further, but we think that energy consumption will be able to reduce it more by allowing path planning considering energy consumption without making a detour when avoiding a collision.

5. Conclusion

In this paper, we proposed a path planning method for drones to reduce energy consumption. In this study, we focus on the energy consumption of the path planning method of drones in the previous study [6], and show two proposed methods for its improvement. In the path planning method using speed

control in the traveling direction proposed in Section 3, speed control in the traveling direction using depth images was added to the path planning method of the drone to improve the problems of the previous study [6]. Also, because AirSim cannot directly acquire energy consumption, the power consumption of the drone was divided into the power consumed by the motor and the power consumed other than the motor in this experiment. power consumption of the motor was calculated from the power of the motor, and power consumption other than the motor was set to a certain value to calculate the power consumption and energy consumption of the drone. The method using deep reinforcement learning proposed in Section 4 uses depth imaging and deep reinforcement learning to control the speed of the drone and avoid collisions. Comparative experiments between the previous study and the two proposed methods show that both of the proposed methods can improve energy consumption compared to the previous study. Specifically, the path planning method using speed control in the traveling direction reduced the arrival time by about 39 seconds and energy consumption by about 42% compared with the previous study. The path planning method using deep reinforcement learning also reduced the arrival time by about 47 seconds and energy consumption by about 30% compared with the previous study. In terms of collision rate, the path planning method using speed control in the traveling direction improved by about 2.8% and the path planning method using deep reinforcement learning improved by about 3.8% compared with the previous study. Comparing the two proposed methods, energy consumption was smaller in the path planning method using speed control in the traveling direction, but the collision rate was smaller in the path planning method using deep reinforcement learning. However, regarding energy consumption, we think that the method using deep reinforcement learning still has room for improvement.

As a future research issue, we plan to experiment by giving not only depth images but also information on the current location and destination of the drone and energy consumption as feature quantities to the input of deep reinforcement learning. We think that this will enable us to carry out path planning considering energy consumption. In this experiment, we focused on the power consumption drone motor and other than the motor, so in future experiments, we think it is necessary to think about a more accurate calculation method of the power consumption on AirSim, to conduct experiments in an environment considering dynamic obstacles, and to conduct experiments using a real vehicle in the real space.

Acknowledgments

This work is partly commissioned by NEDO (Project Number JPNP22006).

Papers

[1] Ministry of Land, Infrastructure, Transport and Tourism, [Online]. Available: https://www.mlit.go.jp/common/001428828.pdf [Accessed On Dec., 2022].

[2] Ministry of Land, Infrastructure, Transport and Tourism, [Online]. Available: https://www.mlit.go.jp/common/001085970.pdf [Accessed On Dec., 2022].

[3] Ministry of Land, Infrastructure, Transport and Tourism, Civil Aviation Bureau : Flight Rules for Unmanned Aircraft(Drones and Model Aircraft, etc.) - MLIT Ministry of Land, Infrastructure, Transport and Tourism. [Online]. Available: https://www.mlit.go.jp/en/koku/uas.html [Accessed On May, 2023].

[4] A. Moffatt, E. Platt, B. Mondragon, A. Kwok, D. Uryeu and S. Bhandari, "Obstacle Detection and Avoidance System for Small UAVs using a LiDAR," In Proc. of International Conference on Unmanned Aircraft Systems, 2020.

[5] E. Perez, A. Winger, A. Tran, C. Garcia-Paredes, N. Run, N. Keti, S. Bhandari and A. Raheja, "Autonomous Collision Avoidance System for a Multicopter using Stereoscopic Vision," In Proc. of International Conference on Unmanned Aircraft Systems, 2018.

[6] T. Shimada, H. Nishikawa, X. Kong and H. Tomiyama, "A Dynamic Path Planning Method for Multirotor Using Depth Images in AirSim," In Proc. of International Workshop on Nonlinear Circuits, Communications and Signal Processing, 2021.

[7] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,” In Proc. of Field and Service Robotics, 2017.
[8] S. Wang, J. Chen, Z. Zhang, G. Wang, Y. Tan and Y. Zheng, "Construction of a virtual reality platform for UAV deep learning," In Proc. of Chinese Automation Congress, 2017.
[9] C. Y. Ho, S. Y. Tseng, C. F. Lai, M. S. Wang and C. J. Chen, "A Parameter Sharing Method for Reinforcement Learning Model between AirSim and UAVs," In Proc. of International Cognitive Cities Conference, 2018.
[10] S. Song, Y. Zhang, X. Qin, K. Saunders and J. Liu, "Vision-guided Collision Avoidance Through Deep Reinforcement Learning," In Proc. of IEEE National Aerospace and Electronics Conference, 2021.
[11] S. Ouahouah, M. Bagaa, J. Prados-Garzon and T. Taleb, "Deep-Reinforcement-Learning-Based Collision Avoidance in UAV Environment," IEEE Internet of Things Journal, vol. 9, no. 6, pp. 4015-4030, 2022.
[12] E. Çetin, C. Barrado, G. Muñoz, M. Macias and E. Pastor, "Drone Navigation and Avoidance of Obstacles Through Deep Reinforcement Learning," In Proc. of Digital Avionics Systems Conference, 2019.
[13] Intel Corporation, Intel RealSenseTM Camera 400 Series Product Family Datasheet. [Online]. Available: https://www.intelrealsense.com/wp-content/uploads/2022/11/Intel-RealSense-D400-Series-Datasheet-November-2022.pdf?_ga=2.152401273.1318330412.1674320733-931204647.1665898695 [Accessed On Sep., 2022].
[14] NVIDIA Corporation, NVIDIA Jetson Orin Nano Developer Kit. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/ [Accessed On May, 2023].
[15] Microsoft, Releases - microsoft/AirSim. [Online]. Available: https://github.com/microsoft/AirSim/releases [Accessed On May, 2022].
[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms,” 2017.
[17] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus and Noah Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," Journal of Machine Learning Research, vol. 22, no. 268, pp. 1-8, 2021.
[18] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba, "OpenAI Gym," 2016.