Automatic Path Planning for Spraying Drones Based on Deep Q-Learning

*Ya-Yu Huang, Zi-Wen Li, Chun-Hao Yang, Yueh-Min Huang**

Department of Engineering Science, National Cheng Kung University, Taiwan
n96084141@gs.ncku.edu.tw, n96101147@gs.ncku.edu.tw, n96091601@gs.ncku.edu.tw, huang@mail.ncku.edu.tw

Abstract

The reduction of the agricultural workforce due to the rapid development of technology has resulted in labor shortages. Agricultural mechanization, such as drone use for pesticide spraying, can solve this problem. However, the terrain, culture, and operational limitations in mountainous orchards in Taiwan make pesticide spraying challenging. By combining reinforcement learning with deep neural networks, we propose to train drones to avoid obstacles and find optimal paths for pesticide spraying that reduce operational difficulties, pesticide costs, and battery consumption. We experimented with different reward mechanisms, neural network depths, flight direction granularities, and environments to devise a plan suitable for sloping orchards. Reinforcement learning is more effective than traditional algorithms for solving path planning in complex environments.

Keywords: Deep Q-Learning (DQN), Drone (UAV), Reinforcement learning (RL), Path algorithm, Smart agriculture

1 Introduction

The number of people employed in the technology industry has increased rapidly in the past few years, resulting in fewer people working in agriculture. In this situation, smart agriculture is a critical technology for maintaining the yield and quality of crops [1-5]. Spraying drones have now been added to pesticide spraying technology [6-7]. This technology reduces the burden of carrying pesticide tanks and eliminates the need for farmers to be exposed to a pesticide-filled environment, thus enhancing the effectiveness of pest control.

The technology for spraying drones has many problems yet to be solved. When using a crop spraying drone for plant protection in orchards, operating environments include complexities such as power equipment towers, nylon nets for agricultural production management, buildings, and so on [8-10]. Also, most orchards are located on hillsides with different slopes and trees growing at different heights [11]. In addition, many farmers plant a variety of crops on the slope to reduce costs. At present, spraying drones are generally still operated manually. These problems complicate the task of drone pilots [12], and out-of-sight obstacles also make it impossible for the pilot to make immediate judgments to avoid the obstacles, resulting in aircraft accidents [13]. It is also difficult to spray different pesticides on different crops [14].

The above problems could be solved if the spraying drones could automatically spray after flying to the destination given the input position. In this paper we seek to implement a drone path planning system to automatically decide the best pesticide spraying route based on the current environmental parameters of the orchard, accounting for topography, tree density, pest prevalence, and so on by using the proposed path algorithm. Since current route planning requires human judgment on the necessary points to be taken, route algorithms for the traveling salesman problem (TSP) are suitable for shortest path planning [15]. Although the system is cost-effective, dynamic decisions cannot account for all the different environmental factors; the system automatically determines the critical points and plans the best path according to the different weights.

Reinforcement learning (RL) [16] is a type of machine learning that takes the actions that optimize the benefits based on the current environment [17-18]. A classic example is the obstacle avoidance system used in robot vacuums [19]. In this paper we adopt the deep Q-learning (DQN) approach, which combines deep learning with RL [20]. On slopes with hilly terrain, we take into consideration the target crop density, the slope height, and the location of pest infestations to find the most efficient pesticide spraying path to reduce pesticide use and UAV battery usage.

The rest of this study is organized as follows. In Section 2, we briefly survey research related to drone obstacle avoidance and reinforcement learning. In Section 3, we outline our actual observations in a sloping orchard and introduce the test platform and environmental data used in the experiments. Section 4 introduces the research methodology, including the use of DQN in RL as a neural network architecture, and the environmental parameter settings to train the best neural network and output location for pesticide spraying paths. In Section 5, we experiment with different parameters and analyze the various results. Finally, in Section 6, we conclude with a discussion on the results of Section 5.

2 Related Work

This section covers recent research on UAV obstacle avoidance applications, explains the basic solution of TSP in path planning, introduces reinforcement learning. and discusses related applied techniques.

*Corresponding Author: Yueh-Min Huang; E-mail: huang@mail.ncku.edu.tw
DOI: 10.53106/160792642023052403001

2.1 Applications of Drone Obstacle Avoidance

Much research has been conducted on RL-based drone obstacle avoidance [21-26]. For example, Mnih et al. [27] propose a drone path planning method based on deep reinforcement learning (DRL) applied to an extensive task scenario. Their study focuses on coverage path planning (CPP), where the UAV investigates areas of interest for data harvesting using distributed IoT sensor devices. Given the information from these environments, two double DQNs (DDQN) [28] of the same architecture are trained to enable the UAVs to determine their flight paths in very different task scenarios. To improve the efficiency of the path planning method, Liu et al. [29] use DQN for goal tracking. They treat the UAV as an agent that seeks a safe and efficient way to fly. The authors approach this as a Markov decision process (MDP) [30] problem. DQN is effective even in unknown environments.

While the above is a static environment simulation, drones are often used in dynamic environments with potential threats. Yan et al. [31] use a DRL approach for drone path planning, evaluating the survival probability of UAVs under enemy radar detection and missile attack in a software simulation. The study uses dueling double DQNs (D3QN) [27, 32] to model a set of environment maps as input and computes the corresponding candidate actions. They then select an operation via the epsilon-greedy strategy and heuristic search rule. This method yields good results on both static and dynamic tasks.

2.2 Traveling Salesman Problem

TSP is a relatively complex non-deterministic polynomial (NP) problem [33], mainly built in the context shown in Figure 1. For a traveler who seeks to travel to seven cities and knows the distance between each city and the other cities, the computer calculates the shortest route from the starting city to visit each city without visiting any more than once [34].

The traditional approach to solving the TSP problem is the brute-force method, where all possible combinations of routes are arranged to find the optimal solution [35]. However, this significantly reduces the computational efficiency and incurs high time complexity with an increased the number of points. Thus path algorithms have been developed to solve the TSP problem [36], for example, the ant colony optimization (ACO) [37], particle swarm optimization (PSO) [38-39], simulated annealing (SA) [40], genetic algorithms (GA) [41], and so on. All of these use a path algorithm to calculate the shortest path under the condition that each point is visited only once.

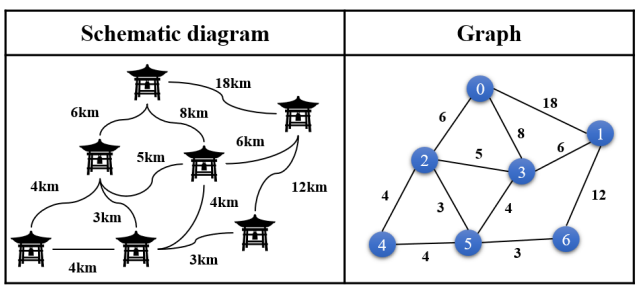

Figure 1. Traveling salesman problem

2.3 Reinforcement Learning

Reinforcement learning [16] is a type of machine learning concerned with how intelligent agents ought to take action in a complex environment to maximize the cumulative reward [42]. RL agents adopt a trial-and-error approach when playing games, and the agent is rewarded or punished for each action performed. Without any hints or advice about the game, the agent decides how to perform the task to maximize its reward. This trial-and-error learning approach is used to achieve the desired end goal. The following is an introduction to several important RL concepts, as well as their applications and limitations.

2.3.1 Markov Decision Process (MDP)

A Markov decision process (MDP) [30] is an important concept in RL which was originally used in probability theory and statistics. MDPs can be used to learn how to make decisions through mathematical algorithms when faced with a situation that requires random choices. Current MDP applications in optimization research include dynamic planning [43] and RL [44].

An MDP can be described by the following mathematical equation, with the basic requirements:

1. Environment states (each frame of the map)
2. Starting state (the agent's starting position)
3. Actions that the agent can perform (up, down, left, right)
4. Probability of selecting an action (probability of going to a different frame)
5. Reward for reaching a specific state (points are deducted when encountering traps)

Given these settings, $T(s, a, s')$ is used to denote the probability of starting in state , taking action a , and ending up in state s'. This probability is multiplied by the reward for state s' and the expected benefit of the Q-state. The largest value of $Q(s, a)$ depends on which state s' that $Q(s, a)$ ends up in, as in

$$Q(s,a) = \sum_{s'} T(s,a,s')[R + \gamma V(s')]. \quad \textbf{(1)}$$

Despite their effectiveness, MDPs have their limitations. For example, MDP training considers only rewards but not the cost of the agent's actions. It is however crucial to consider the cost of the actions taken by the agent and be cautious about the next step.

2.3.2 Q-Learning

Q-Learning [45] is a value-based RL algorithm that updates the value function according to the Bellman equation. The value of each state is estimated precisely, and the cumulative reward of continuous behavior over time is considered instead of only the reward from the current action. This long-term reward is denoted as a Q-value, where "Q" represents quality, which indicates the usefulness of the action for obtaining future rewards; the Q-value for each state and corresponding action is

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a)\right). \quad \textbf{(2)}$$

In the Q-learning process, the $Q(s_t, a_t)$ value is randomly initialized. The initial state s value in each iteration as well as the reward r and value s' of the following state are obtained after each step is chosen according to the probability of Q to perform action a. Then the Q-function is recalculated using Equation (2) and the new values are updated in the Q-table, and its Q-value is updated according to the selection behaviors of each state. After the action is executed, the state is current, and the above steps are repeated until the endpoint.

2.3.3 Deep Q Network (DQN)

In a Q-table implementation, a Q-table is a table that stores Q values and is indexed by state and action. However, this indexed table can be created only if the number of states and actions is small enough. Deep Q-learning [46] instead uses a deep neural network to extract features and approximate the desired Q function.

In deep Q-learning, the neural network is fed with the input-output pair to eventually approximate the function of the input-output correspondence, which is transformed into a more trainable form of the π(State) = Action policy. The advantage of replacing the Q-table with a neural network is that the neural network automatically extracts features from the vast state space with different variants.

3 Experimental Environment

In this section, we introduce our team's field observation of the environmental conditions in Taiwan's mountain orchards to understand what obstacles drones will encounter when flying in mountainous areas. Then we introduce the system environment. In this study, as our main focus is on real-world augmented learning combined with deep learning, we use TensorFlow to build deep Q-learning training models.

3.1 Mountain Orchard Environment

Our research team observed fruit trees in the mountains of Taiwan in the field, and in other research on precise pesticide spraying by drones [47], we visited the sloping orchards of Nanhua in Tainan, Taiwan. Figure 2 is an aerial photo of an orchard in Nanhua, Taiwan: the orchard contains various structures and buildings, and the height and size of the trees vary widely, as well as the species of the trees. These factors all complicate the task of drone pilots. Figure 3 is a side view of the orchard in Nanhua, Taiwan, where we flew the drone. We observe a hilly slope, which also constitutes a challenge for drone pilots.

Figure 2. Aerial photo of orchard in Nanhua, Taiwan

Figure 3. Side view of orchard

3.2 Software Environment

TensorFlow [48], an open-source platform for building machine learning applications, was developed by the Google Brain team and first made public in late 2015 under the Apache Open Source license. It is a symbolic mathematical library that uses DataStream and differentiable programming to perform training and inference tasks focused on deep neural networks. Thus, TensorFlow sets up DataStream graphs and structures as shown in the TensorFlow architecture in Figure 4, where the inputs are treated as multidimensional arrays called Tensors to define how the data moves in the graph. The architecture is divided into three main parts: data preprocessing, model building, and model training and evaluation. Complex programming is not required to build, configure, or program neural networks, which makes it easier to research machine learning and neural networks. TensorFlow also has an important feature called TensorBoard, which monitors TensorFlow operations graphically and visually.

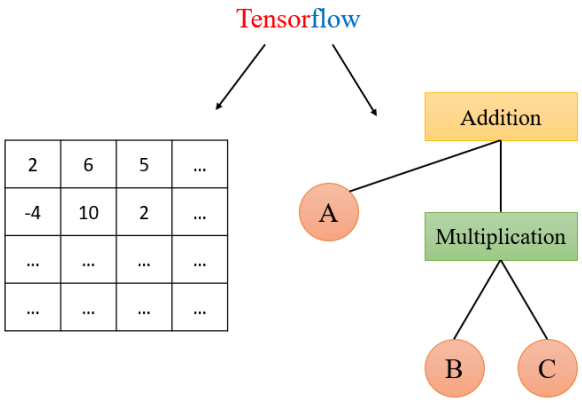

Figure 4. TensorFlow architecture

3.3 Interface Environment

The environment interface is designed mainly using the PyQt5 suite [49]. Qt is a cross-platform GUI framework that supports Windows, MacOS, Linux, and other major operating systems. The primary programming language used is C++, which is relatively difficult to write and install. Therefore, for this study we implemented the environment interface of the map in PyQt5 and generated a spreadsheet with reward values as RL training data, as shown in Figure 5.

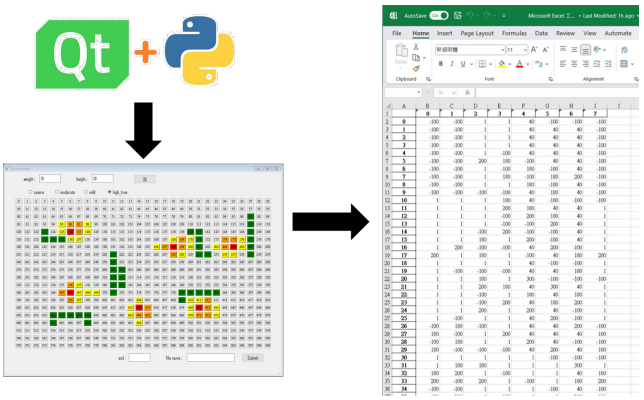

Figure 5. Use PyQt5 to create the environment map for this study and generate the reward data file

4 System Design and Implementation

This section introduces the DQN implementation for the RL drone application. After setting up the environment and creating an environment map, we generated the reward data as a training sample for DQN to decide the flight direction of the drone, after which we processed the data and output a drone flight path.

4.1 System Architecture

The system architecture here is divided into two parts: First, we discuss the augmented learning in the UAV automatic path planning system. Second, we describe the experimentation steps.

4.1.1 System Architecture for UAV Path Planning

We developed an unmanned path intelligence system, as shown in Figure 6. In the area to spray with pesticides, the critical factors were the height of the tree tops, the density of the target species, and the severity of the pest infestation. After integrating these data, we planned a PyQt5-based environment map interface. Finally, we generated the rewards table needed for deep Q-learning. The reward table allows the neural network to learn the suitability of each flight direction during training. If a direction is not suitable, a penalty is given; the more suitable the direction, the higher the reward. The weights are stored after the whole training is completed.

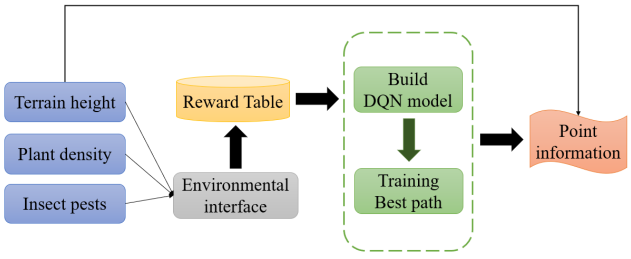

Figure 6. Overall UAV automatic path planning system architecture

4.1.2 DRL Model Training Steps

Traditional path algorithms require information about the flight points before calculating the shortest path, but that is not suitable for complex environments. In this study, DRL is used for suitable path planning through the DQN neural network architecture. Figure 7 describes the process of training the neural network model. The drone is considered an agent, and during the training of the DQN model, the information about the flight direction (state) at each position is obtained from the environment map in the yellow block on the right, which is then input to the DQN model and used as a training sample. The reward value is updated to reflect the different conditions to determine which direction is the best choice, and finally, an optimal path is obtained.

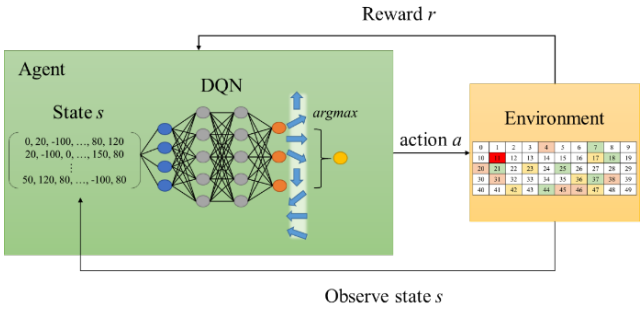

Figure 7. Nerve network model training steps

4.2 Environmental Data Sample

The computer uses environmental data to determine the priority of spraying and sequencing flight paths to plan the best paths for drones. The three main elements of this study are the areas where the lychee giant stink bugs (*Tessaratoma papillosa*) concentrate, the density of the lychee and longan trees, and the height of the slope. Below, we present the coordinates of these three elements.

Liu [29] uses a D3QN-based DRL approach to train drones for IoT green mobility management to collect environmental sensing data. To ensure immediate transmission with minimum delay, each area is assigned a different color based on the data constraint delay and energy consumption to prioritize the data, as shown in Figure 8. We follow this work in assigning different colors to different grids to reflect the priority ranking when the environmental map information is compiled.

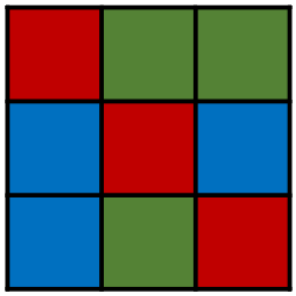

Figure 8. Color-based area prioritization [29]

Figure 9 shows the environmental interface of this study. We classify the environment and divide the spraying area into a grid of 20x30 cells, and then manually input the environmental information. The environmental messages are divided into four colors. Red, orange, and yellow represent

the degree of pest infestation or the high-, medium-, and low-density target tree species, respectively. During the experiment, we observed that in areas with severe pest infestation, the pests often fly to neighboring areas. Therefore, orange and yellow areas are adjacent to red areas. In addition, to prevent drone collisions, areas with high obstacles such as high treetops or utility poles are marked green. After thus, we input the final landing location and the reward file name and generate a training sample.

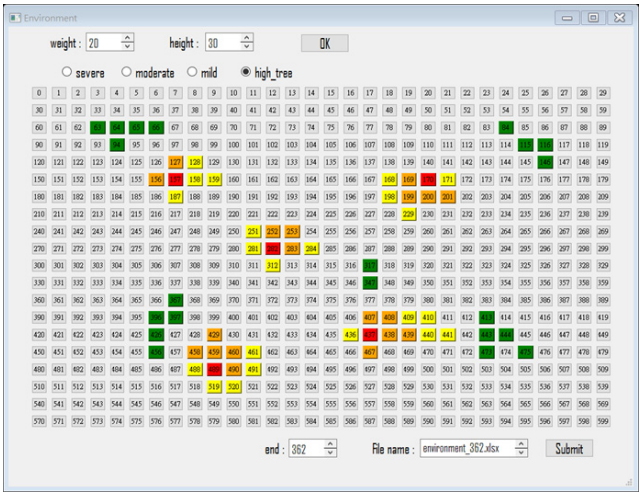

Figure 9. Interface of study environment

4.3 Data Processing and Model Implementation

4.3.1 Reward Table Generation

The environmental interface shown in Figure 10 prioritizes the different cells in the target agricultural spraying area with trial-and-error and suggestions from agricultural experts. In this study, the drone moved in eight directions, as shown in Figure 11, with a corresponding reward for each direction. The program automatically generated reward files for all grids containing each direction of movement. Assuming the area was divided into 20x30 cells, there were 600 cells, each with eight movement directions, thus requiring the generation of 600x8 rewards, as shown in the rewards table in Figure 11.

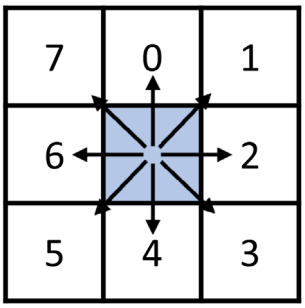

Figure 10. Eight numbered directions of drone movement

	0	1	2	3	4	5	6	7
0	-200	-200	20	20	-100	-200	-200	-200
1	-200	-200	-100	120	20	-100	1	-200
2	-200	-200	1	1	120	20	20	-200
3	-200	-200	1	1	20	120	-100	-200
4	-200	-200	80	150	20	20	20	-200
5	-200	-200	1	120	150	20	20	-200
6	-200	-200	1	1	120	150	80	-200
7	-200	-200	-100	-100	20	120	20	-200
8	-200	-200	1	1	-100	20	20	-200
9	-200	-200	-200	-200	20	-100	-100	-200
10	1	1	20	20	-100	-200	-200	-200
11	1	-100	120	150	20	-100	-100	1
12	-100	1	1	120	150	20	20	1
13	1	1	1	1	120	150	120	-100
14	1	80	150	1	20	120	20	1
15	80	1	120	80	20	20	20	1
16	1	1	1	1	80	20	150	80
17	1	-100	-100	1	20	80	120	1
18	-100	1	1	1	20	20	20	1
19	1	-200	-200	-200	20	20	-100	-100
20	-100	1	20	200	20	-200	-200	-200
21	1	120	150	80	200	1	-100	-100
22	120	1	120	1	80	200	20	1
23	1	1	1	1	20	80	150	120
24	1	150	1	1	20	20	120	1
25	150	120	80	120	20	20	20	1

Figure 11. Reward table
(The horizontal "action" axis represents the movement direction, and the vertical "state" axis represents each cell)

Different values of the reward value were experimented with, resulting in the reward parameter configuration shown in Table 1. As repeated paths increased the power consumption of the aircraft, paths for such iterations were zeroed and relearned. The reward values were set to 150, 120, and 80 for areas requiring pesticide spraying, in decreasing order of priority. The reward value was set to 50 for directions towards the endpoint and 200 for the area adjacent to the endpoint to ensure that the DQN model prioritized the shortest path during training to reduce the power consumption and avoid spraying pesticides on unnecessary areas.

Table 1. Reward parameters (see Experiment 3 in Section 5.1)

	Color	Reward
Outside boundary		-10
Higher terrain areas	Green	-10
Severe pest infestation	Red	150
Moderate pest infestation	Orange	120
Slight pest infestation	Yellow	80
Endpoint		200
In direction of endpoint		50

4.3.2 Gradient Descent

In optimization theory, gradient descent [50] is a first-order method for finding the best solution. Gradient descent is used to find the local minimum of the objective function; because the gradient points toward the local maximum, gradient descent proceeds in the opposite direction of the gradient.

As gradient descent continually updates the parameters to find the solution, a random set of solutions of the initial parameters is first generated. Then, according to these randomly generated solutions, the gradient direction of the solution size is calculated and the solution is subtracted from the gradient direction. Where x is the first update parameter, t is the iteration number, and γ is the learning rate, the formula is

$$x^{(t+1)} = xs^{(t)} - \nabla f(x^{(t)}). \quad (3)$$

4.3.3 Experiment Replay

Experiment replay [51] refers to learning a random mini-batch during training by storage-sampling, which prevents the neural network from falling into an optimal local situation in the case of sequential access. DQN uses Experience Replay so that the agent does not need to be trained after each step. In this paper, Experience Replay is used to accelerate the training of DQN by performing small batch training every ten steps. In this paper, we use Experience Replay to accelerate the training of DQN by conducting small batch training every 10 steps, and the DQN is divided into the following steps by the learning method of Experience Replay (as shown in Figure 12):

1. Pre-process and provide the environment (state s) to the DQN, which returns the Q values of all possible actions in the state.
2. Select an action using the epsilon-greedy strategy. With probability epsilon, select a random action a; otherwise, select the action with the maximum Q value, e.g., $<s, a, r, s'> a = argmax\ (Q\ (s, a, w))$.
3. Perform the action in state , and then move to the new state to collect the reward. State s' is the image preprocessing of the next screen. This data is converted and stored in the replay buffer as $<s, a, r, s'>$
4. Sample random batches of conversion records from the replay buffer, which calculates the loss.
5. Calculate the squared difference between the target and predicted Q as

$$Loss = (r + \gamma max_{a'} Q(s', a';\ \theta) - Q(s, a;\ \theta))^2. \quad (4)$$

6. Minimize the loss by gradient descent on the network parameters.
7. After each C iterations, copy the network weights to the target network.
8. Repeat these steps for M levels.

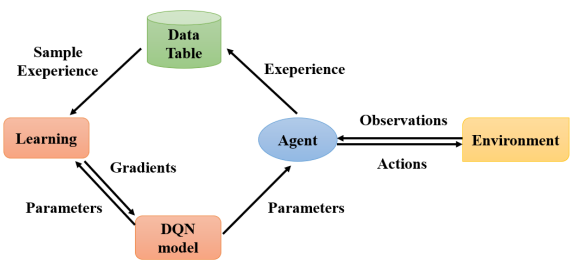

Figure 12. DQN training

5 Experiment Results and Analysis

Here we implement path planning for the augmented DQN algorithm and analyze the results.

5.1 Results for Different Rewards

RL [17] uses a reward mechanism to enable the agent to analyze actions and generate feedback, training the agent to learn independently. As the design of the rewards greatly influences the training model, we attempted to determine the best reward mechanism. The 15x20 simulation environment of this experiment is shown in Figure 13. In the initial 2000 training iterations, the agent chose actions randomly from up, down, left, and right, and the feedback value of each action was recorded using the ER mechanism. With this experience from the random learning, we proceeded with 28000 more training iterations. The experimental trend is presented in Table 2 with three different reward parameters, as shown in Figure 14 to Figure 16, where the horizontal axis is the number of iterations, and the vertical axis is the RL cost parameter. Because the input data in DQN changes at every step, the cost curve exhibits clear oscillation depending on the learning situation. The resultant trend differs from that of the loss function in deep learning, which decreases steadily, but we still observe a gradual smoothing trend.

Figure 13. Simulation environment used for reward experiment

Table 2. Reward parameters

Reward	Experiment 1	Experiment 2	Experiment 3
Outside the boundary	-10	-10	-10
Higher terrain areas	-10	-10	-10
Severe area	15	80	150
Moderate area	12	40	120
Slight Area	8	20	80
Endpoint	20	100	200
Direction to the end point	4	10	50

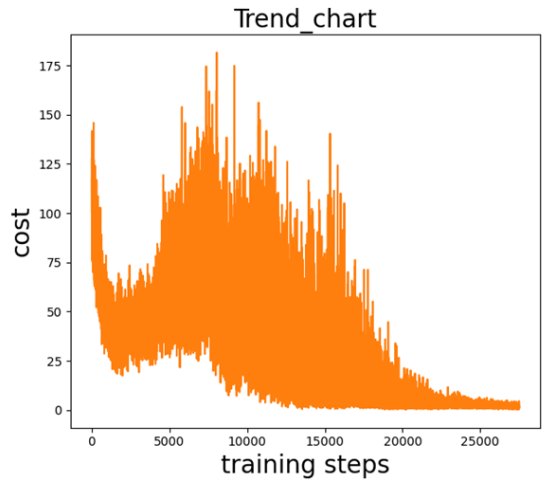

Figure 14. Experiment 1 cost trend

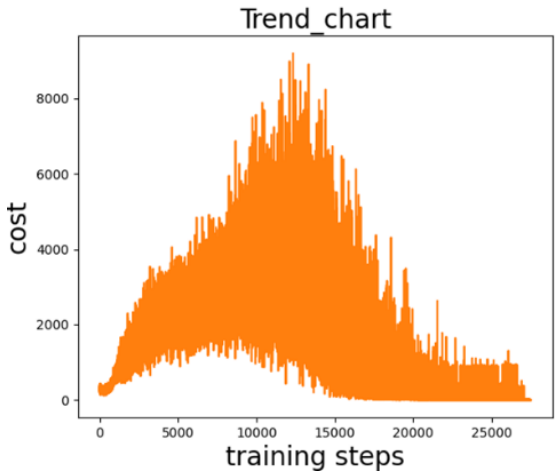

Figure 15. Experiment 2 cost trend

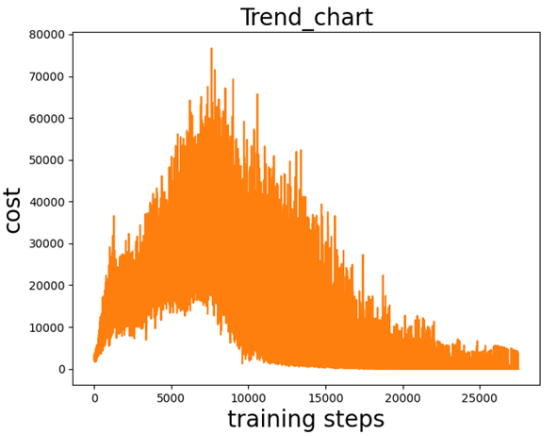

Figure 16. Experiment 3 cost trend

The above results show that the clearer the reward and punishment, the smoother the cost curve. Training time, which is little affected by this, is about 400 seconds. If the map is enlarged, the range of the rewards and penalties should be increased accordingly to produce accurate experimental results. Excessive differences between rewards and penalties may be counterproductive and cause the model to ignore slight or moderate areas of infestation. Thus the grid size should also be considered when setting the reward value. For example, the simulation environment in this experiment was 15x20, or 300 cells, and the reward value was set between 300 and -100.

5.2 Results for Different Network Levels

In this study, we used an RNN network as the DQN architecture. The input layer reads the environmental information, passes it to the hidden layer for training, uses backpropagation to tune the internal parameters, and produces the output probabilities for each action in the output layer, from which the agent takes the action with the maximum probability, that is, the UAV's flight direction.

Here we compare the number of hidden layers in the neural network to understand the influence of the layers. We used a 20x30 grid for the simulation, and used the data from Experiment 3 in Table 2 for the reward design after changing the network layer structure. According to the experimental results in Figure 17 and Figure 19, the two-layer neural network considers more environmental factors, whereas the three-layer neural network has a shorter path. In terms of cost, the three-layer neural network in Figure 20 is smoother than the two-layer network in Figure 18. This result may be because the three-layer neural network has too many parameters, resulting in overfitting. Therefore, we believe that the two-layer neural network is more consistent with optimal path planning, and the three-layer neural network is more consistent with shortest path planning. Since our goal is optimal path planning, the two-layer neural network is better.

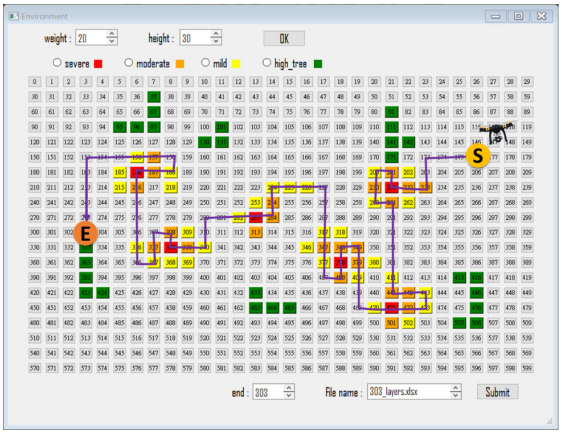

Figure 17. Optimal path planning for two-layer neural network (S: start; E: end)

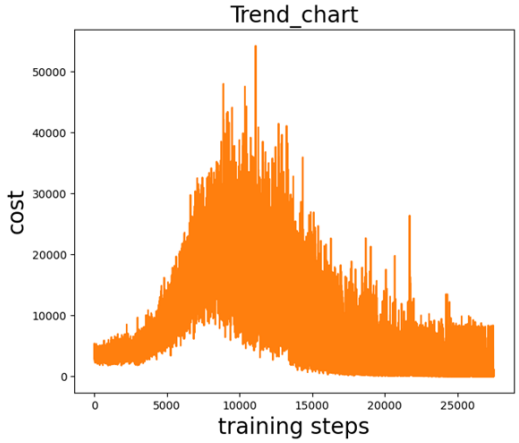

Figure 18. Cost trend of two-layer neural network

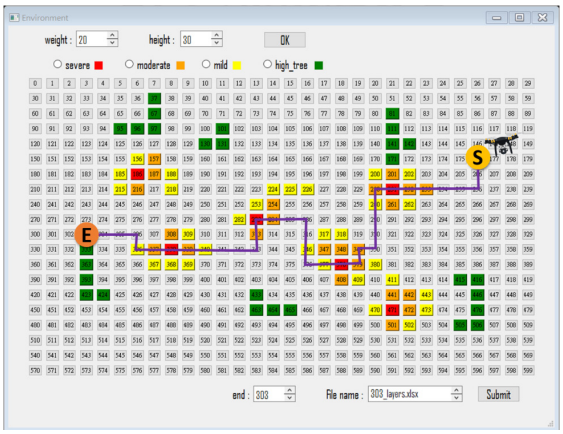

Figure 19. Optimal path planning for three-layer neural network

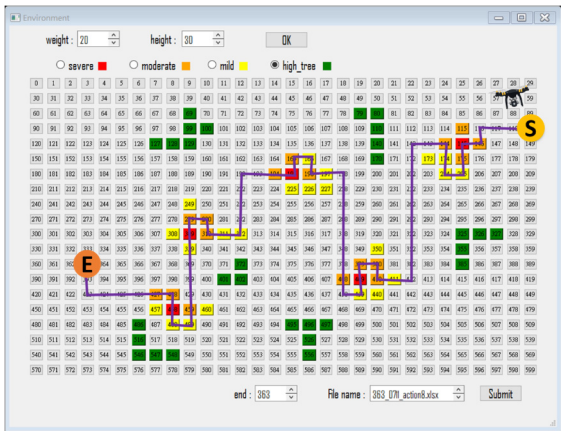

Figure 21. Optimal path for four directions

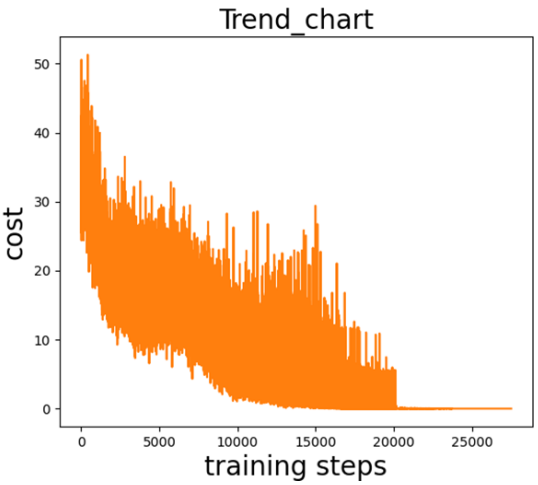

Figure 20. Cost trend of three-layer neural network

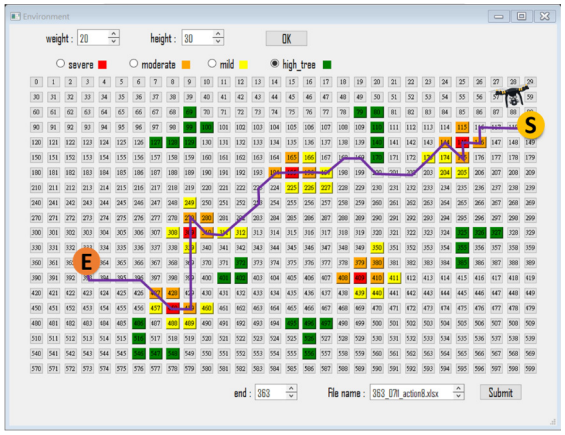

Figure 22. Optimal path for eight directions

5.3 Flight Direction Results

For the path planning of the spraying drone, to consider cost issues such as battery and drug quantity, we investigated whether eight directions (up, top left, left, bottom left, bottom, bottom right, right, and top right) yields more efficient paths and hence reduce costs compared with the more simpler four directions (up, left, bottom, and right).

For this experiment, we used a 20x30 simulation and adjusted the model's output layer to compare the experimental results. In Figure 21 and Figure 22, with four directions, the surrounding area—the orange areas with medium infestation and the yellow areas with slight infestation—is visited and sprayed with higher point coverage. However, eight directions predominantly cover the red areas with severe infestations; they cover less of the orange and yellow areas and focus more on generating shorter paths. Therefore, we believe four directions is better than eight for large-area applications.

5.4 Results for Different Environments

In this study, the simulated environment was a sloping orchard characterized by hilly terrain, with many obstacles to flying drones. The tree species to be sprayed were scattered, and pest tended to concentrate around the target tree species. These are all factors to consider when planning a simulation path. Here we compare different environmental factors. First, we differentiate by the density of tree species alone. Red, orange, and yellow correspond to areas with high, medium, and low tree density. In this case, because tree density is unrelated to the degree of pest infestation, these appear in Figure 23 as single points of dispersion; the experimental results show that the system ignores minor cells. We also find that the cost is higher when using only four flight directions. Therefore, when only the density of the tree species is important, eight flight directions is more effective.

In Figure 24, we consider the level of pest infestation. Although some non-red cells are ignored, the path is more effective overall. In Figure 25, after adding the terrain height, the system drops one or two red cells that should be sprayed. These results show that in complex experimental sites such as these, terrain and other factors block areas that the drones should have visited, which further affects the path planning results.

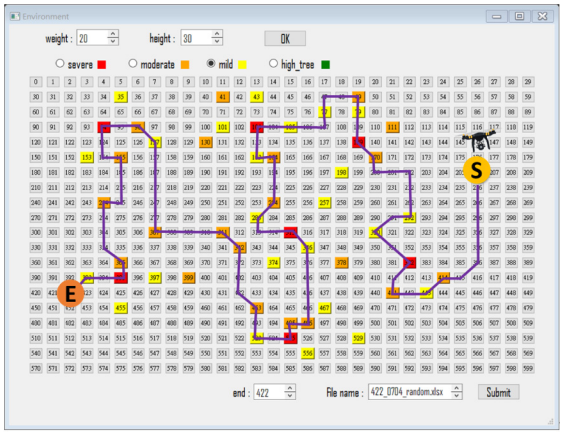

Figure 23. Optimal plan considering only tree density

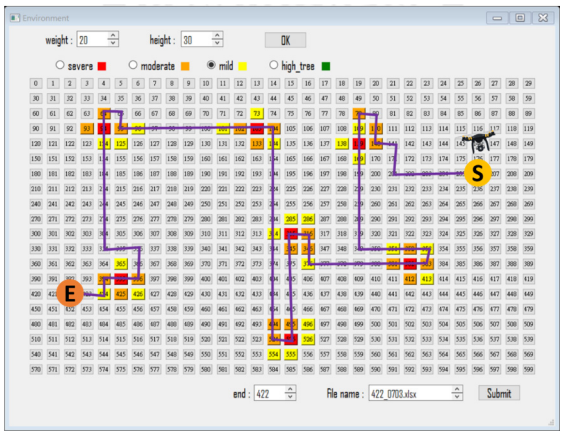

Figure 24. Optimal plan considering tree species density and pest-infestation locations

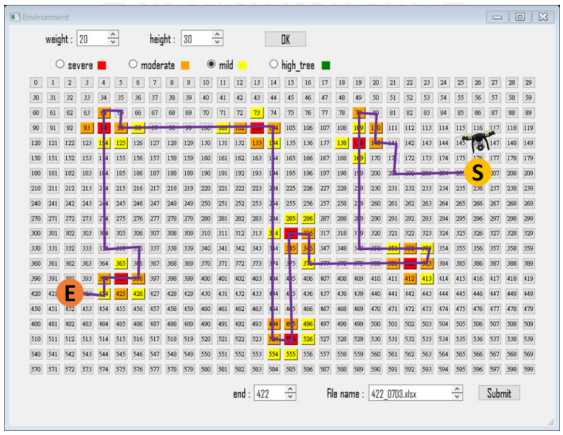

Figure 25. Optimal plan considering tree species density, pest-infestation locations, and terrain height

6 Conclusion

We propose a deep reinforcement learning method based on DQN for use in drone path planning. We seek to significantly reduce the cost of pesticides and to leverage smart agriculture by spraying pesticides on only a few target species. By training the model with known environmental factors, the drone flies automatically and avoids obstacles in complex environments. To devise a plan that accounts for an environment with fruit trees on a slope, we experimented with different reward mechanisms, neural network depths, flight direction granularities, and environments to train our DQN model.

In highly complex environments, some target areas are neglected. Sometimes, when the system bypasses higher obstacles, it takes the long way around, causing it to diverge from the optimal planned path and skipping areas that should be sprayed, instead choosing the shortest path. The path planning proposed in this study performs better in simple environments.

Considering that the current training mechanism of the DQN model cannot satisfy all complex situations, one direction for future improvement is adjusting parameters. In addition, our simulations were all two-dimensional: transitioning to three-dimensional simulations would support path planning that better reflects real-world field conditions, as would integrating this study with the UAV flight control system. Thus, after importing the path, the pilot could set parameters such as the hover time at each point and adjust the flight height according to the drone type and spray radius. This would make the agricultural spraying operation more convenient, and constitute truly smart agriculture.

References

[1] P. K. R. Maddikunta, S. Hakak, M. Alazab, S. Bhattacharya, T. R. Gadekallu, W. Z. Khan, Q.-V. Pham, Unmanned aerial vehicles in smart agriculture: Applications, requirements, and challenges, *IEEE Sensors Journal*, Vol. 21, No. 16, pp. 17608-17619, August, 2021.

[2] A. Walter, R. Finger, R. Huber, N. Buchmann, Smart farming is key to developing sustainable agriculture, *Proceedings of the National Academy of Sciences*, Vol. 114, No. 24, pp. 6148-6150, June, 2017.

[3] R. Dagar, S. Som, S. K. Khatri, Smart farming–IoT in agriculture, *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2018, pp. 1052-1056.

[4] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, X. Wang, Internet of things for the future of smart agriculture: a comprehensive survey of emerging technologies, *IEEE/CAA Journal of Automatica Sinica*, Vol. 8, No. 4, pp. 718-752, April, 2021.

[5] X. Yang, L. Shu, J. Chen, M. A. Ferrag, J. Wu, E. Nurellari, K. Huang, A survey on smart agriculture: Development modes, technologies, and security and privacy challenges, *IEEE/CAA Journal of Automatica Sinica*, Vol. 8, No. 2, pp. 273-302, February, 2021.

[6] U. R. Mogili, B. Deepak, Review on application of drone systems in precision agriculture, *Procedia Computer Science*, Vol. 133, pp. 502-509, April, 2018.

[7] A. Hafeez, M. A. Husain, S. P. Singh, A. Chauhan, M. T. Khan, N. Kumar, A. Chauhan, S. K. Soni, Implementation of drone technology for farm monitoring & pesticide spraying: A review, *Information Processing in Agriculture*, Vol. 10, No. 2, pp. 192-203, January, 2023.

[8] S. Wang, S. Xu, C. Yu, H. Wu, Q. Liu, D. Liu, L. Jin, Y. Zheng, J. Song, X. He, Obstacle Avoidance and Profile Ground Flight Test and Analysis for Plant Protection UAV, *Drones*, Vol. 6, No. 5, Article No. 125, May, 2022.

[9] S. Wolfert, L. Ge, C. Verdouw, M.-J. Bogaardt, Big data in smart farming–a review, *Agricultural Systems*, Vol. 153, pp. 69-80, March, 2017.

[10] N. Nhamo, D. Chikoye, *Smart Technologies for Sustainable Smallholder Agriculture*, Elsevier, 2017.

[11] K. Wu, X. Sun, J. Zhang, F. Chen, Terrain following method of plant protection UAV based on height fusion, *Nongye Jixie Xuebao/Transactions of the Chinese Society of Agricultural Machinery*, Vol. 49, No. 6, pp. 17-23, June, 2018.

[12] G. Cao, Y. Li, F. Nan, D. Liu, C. Chen, J. Zhang, Development and analysis of plant protection UAV flight control system and route planning research, *Nongye Jixie Xuebao/Transactions of the Chinese Society of Agricultural Machinery*, Vol. 51, No. 8, August, pp. 1-16, 2020.

[13] J. N. Yasin, S. A. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, J. Plosila, Unmanned aerial vehicles (UAVs): Collision avoidance systems and approaches, *IEEE Access*, Vol. 8, pp. 105139-105155, June, 2020.

[14] A. P. D. Corte, E. M. da C. Neto, F. E. Rex, D. Souza, A. Behling, M. Mohan, M. N. I. Sanquetta, C. A. Silva, C. Klauberg, C. R. Sanquetta, H. F. P. Veras, D. R. A. de Almeida, G. Prata, A. M. A. Zambrano, J. W. Trautenmüller, A. de Moraes, M. A. Karasinski, E. N. Broadbent, High-Density UAV-LiDAR in an Integrated Crop-Livestock-Forest System: Sampling Forest Inventory or Forest Inventory Based on Individual Tree Detection (ITD), *Drones*, Vol. 6, No. 2, Article No. 48, February, 2022.

[15] Y. Xu, C. Che, A brief review of the intelligent algorithm for traveling salesman problem in UAV route planning, *2019 IEEE 9th international conference on electronics information and emergency communication (ICEIEC)*, Beijing, China, 2019, pp. 1-7.

[16] R. S. Sutton, A. G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[17] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: A survey, *Journal of Artificial Intelligence Research*, Vol. 4, pp. 237-285, May, 1996.

[18] J. Elhachmi, Distributed reinforcement learning for dynamic spectrum allocation in cognitive radio-based internet of things, *IET Networks*, Vol. 11, No. 6, pp. 207-220, November, 2022.

[19] E. Prassler, A. Ritter, C. Schaeffer, P. Fiorini, A short history of cleaning robots, *Autonomous Robots*, Vol. 9, No. 3, pp. 211-226, December, 2000.

[20] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, Deep reinforcement learning: A brief survey, *IEEE Signal Processing Magazine*, Vol. 34, No. 6, pp. 26-38, November, 2017.

[21] K. Wan, X. Gao, Z. Hu, G. Wu, Robust motion control for UAV in dynamic uncertain environments using deep reinforcement learning, *Remote Sensing*, Vol. 12, No. 4, Article No. 640, February, 2020.

[22] J. L. Junell, E.-J. Van Kampen, C. C. de Visser, Q. P. Chu, Reinforcement learning applied to a quadrotor guidance law in autonomous flight, *AIAA Guidance, Navigation, and Control Conference*, Kissimmee, Florida, 2015, Article No. 1990.

[23] W. Luo, Q. Tang, C. Fu, P. Eberhard, Deep-sarsa based multi-UAV path planning and obstacle avoidance in a dynamic environment, in: Y. Tan, Y. Shi, Q. Tang (Eds.), *Advances in Swarm Intelligence. ICSI 2018. Lecture Notes in Computer Science*, Springer, 2018, pp. 102-111.

[24] N. Imanberdiyev, C. Fu, E. Kayacan, I.-M. Chen, Autonomous navigation of UAV by using real-time model-based reinforcement learning, *2016 14th international conference on control, automation, robotics and vision (ICARCV)*, Phuket, Thailand, 2016, pp. 1-6.

[25] L. Abouzaid, H. Elbiaze, E. Sabir, Agile roadmap for application-driven Multi-UAV networks: The case of COVID-19, *IET Networks*, Vol. 11, No. 6, pp. 195-206, November, 2022.

[26] F. Kiani, A. Seyyedabbasi, R. Aliyev, M. A. Shah, M. U. Gulle, 3D path planning method for multi-UAVs inspired by grey wolf algorithms, *Journal of Internet Technology*, Vol. 22, No. 4, pp. 743-755, July, 2021.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, *Nature*, Vol. 518, No. 7540, pp. 529-533, February, 2015.

[28] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30, No. 1, pp. 1-7, March, 2016.

[29] W. Liu, P. Si, E. Sun, M. Li, C. Fang, Y. Zhang, Green mobility management in UAV-assisted IoT based on dueling DQN, *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, 2019, pp. 1-6.

[30] M. L. Puterman, Markov decision processes, in: D. P. Heyman, M. J. Sobel (Eds.), *Handbooks in Operations Research and Management Science*, Vol. 2, Elsevier, 1990, pp. 331-434.

[31] C. Yan, X. Xiang, C. Wang, Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments, *Journal of Intelligent & Robotic Systems*, Vol. 98, No. 2, pp. 297-309, May, 2020.

[32] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, N. Freitas, Dueling network architectures for deep Reinforcement Learning, *International Conference on Machine Learning*, New York, USA, 2016, pp. 1995-2003.

[33] M. M. Flood, The traveling-salesman problem, *Operations Research*, Vol. 4, No. 1, pp. 61-75, February, 1956.

[34] J. D. Little, K. G. Murty, D. W. Sweeney, C. Karel, An algorithm for the traveling salesman problem,

Operations Research, Vol. 11, No. 6, pp. 972-989, November-December, 1963.
[35] X. Geng, Z. Chen, W. Yang, D. Shi, K. Zhao, Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search, *Applied Soft Computing*, Vol. 11, No. 4, pp. 3680-3689, June, 2011.
[36] M. Bellmore, G. L. Nemhauser, The traveling salesman problem: a survey, *Operations Research*, Vol. 16, No. 3, pp. 538-558, May-June, 1968.
[37] W. Wang, G. Tong, Multi-path unequal clustering protocol based on ant colony algorithm in wireless sensor networks, *IET Networks*, Vol. 9, No. 2, pp. 56-63, March, 2020.
[38] I. C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, *Information Processing Letters*, Vol. 85, No. 6, pp. 317-325, March, 2003.
[39] M. D. Phung, Q. P. Ha, Safety-enhanced UAV path planning with spherical vector-based particle swarm optimization, *Applied Soft Computing*, Vol. 107, Article No. 107376, August, 2021.
[40] S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi, Optimization by simulated annealing, *Science*, Vol. 220, No. 4598, pp. 671-680, May, 1983.
[41] D. Whitley, A genetic algorithm tutorial, *Statistics and Computing*, Vol. 4, No. 2, pp. 65-85, June, 1994.
[42] J. Hu, H. Niu, J. Carrasco, B. Lennox, F. Arvin, Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning, *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 12, pp. 14413-14423, December, 2020.
[43] M. Giuliani, S. Galelli, R. Soncini-Sessa, A dimensionality reduction approach for many-objective Markov Decision Processes: Application to a water reservoir operation problem, *Environmental Modelling & Software*, Vol. 57, pp. 101-114, July, 2014.
[44] M. v. Otterlo, M. Wiering, Reinforcement Learning and Markov Decision Processes, in: M. Wiering, M. Otterlo (Eds.), *Reinforcement Learning*, Springer, 2012, pp. 3-42.
[45] C. J. Watkins, P. Dayan, Q-learning, *Machine Learning*, Vol. 8, No. 3-4, pp. 279-292, May, 1992.
[46] J. Fan, Z. Wang, Y. Xie, Z. Yang, A theoretical analysis of deep Q-learning, *Learning for Dynamics and Control*, Online Event, Berkeley, CA, USA, 2020, pp. 486-489.
[47] C.-J. Chen, Y.-Y. Huang, Y.-S. Li, Y.-C. Chen, C.-Y. Chang, Y.-M. Huang, Identification of fruit tree pests with deep learning on embedded drone to achieve accurate pesticide spraying, *IEEE Access*, Vol. 9, pp. 21986-21997, February, 2021.
[48] Google, TensorFlow, https://www.tensorflow.org/, accessed July, 2021.
[49] J. Willman, *Modern PyQt*, Apress, 2021.
[50] S. Ruder, An overview of gradient descent optimization algorithms, September, 2016, https://arxiv.org/abs/1609.04747.
[51] L.-J. Lin, *Reinforcement learning for robots using neural networks*, Carnegie Mellon University, 1992.

Biographies

Ya-Yu Huang worked as a research assistant in the Knowledge, Information and Database Systems Laboratory from 2018 to 2020. She obtained a Master's degree in Engineering Science from National Cheng Kung University in 2021. Her research interests include the application of remote sensing in agriculture, machine learning in satellite imagery, optical satellites and aerial images, and image processing.

Zi-Wen Li received his bachelor's degree in Information Engineering from Feng Chia University in 2021. He is currently a master's student in the Department of Engineering Science at National Cheng Kung University. His research topics include applications in smart agriculture, drone path planning, embedded systems, IoT, machine learning, E-Learning, and web development.

Chun-Hao Yang received the B.S. degree in Electrical Engineering from Tatung University, Taiwan, in 2019. He is currently a master student with the Department of Engineering Science, National Cheng Kung University, Taiwan. His research interests include machine learning in aerial imagery, remote sensing in agriculture, embedded system and IoT.

Yueh-Min Huang is a Chair Professor in Department of Engineering Science and Institute of Education, National Cheng-Kung University, Taiwan. His research interests include e-Learning, multimedia communications, and artificial intelligence. He received his MS and Ph.D. degrees in Electrical Engineering from the University of Arizona in 1988 and 1991 respectively. He has co-authored 3 books and has published more than 280 refereed journal research papers. Dr. Huang has received many research awards, such as Taiwan's National Outstanding Research Award in 2011/2014, as well as 2017 Taiwan Outstanding IT Elite Award. He has completed over 60 Ph.D. and 300 MS thesis students. Dr. Huang is in the editorial board of several international journals in the area of educational technology, computer communications, and web intelligence. Dr. Huang is also the funding chair of International Symposium of Emerging Technologies for Education (SETE) and International Conference of Innovative Technologies and Learning (ICITL). Dr. Huang is a senior member of the IEEE and became Fellow of British Computer Society in 2011.