biomimetics

Article

E-DQN-Based Path Planning Method for Drones in Airsim Simulator under Unknown Environment

Yixun Chao [1], Rüdiger Dillmann [2], Arne Roennau [2] and Zhi Xiong [1,*]

[1] Navigation Research Center, School of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[2] FZI Research Center for Information Technology, 76131 Karlsruhe, Germany
* Correspondence: xiongzhi@nuaa.edu.cn

Abstract: To improve the rapidity of path planning for drones in unknown environments, a new bio-inspired path planning method using E-DQN (event-based deep *Q*-network), referring to introducing event stream to reinforcement learning network, is proposed. Firstly, event data are collected through an airsim simulator for environmental perception, and an auto-encoder is presented to extract data features and generate event weights. Then, event weights are input into DQN (deep *Q*-network) to choose the action of the next step. Finally, simulation and verification experiments are conducted in a virtual obstacle environment built with an unreal engine and airsim. The experiment results show that the proposed algorithm is adaptable for drones to find the goal in unknown environments and can improve the rapidity of path planning compared with that of commonly used methods.

Keywords: biological inspiration; E-DQN; reinforcement learning; drone; airsim; unreal engine

check for updates

Citation: Chao, Y.; Dillmann, R.; Roennau, A.; Xiong, Z. E-DQN-Based Path Planning Method for Drones in Airsim Simulator under Unknown Environment. *Biomimetics* **2024**, *9*, 238. https://doi.org/10.3390/biomimetics9040238

Academic Editor: Yongquan Zhou

Received: 30 December 2023
Revised: 13 March 2024
Accepted: 12 April 2024
Published: 16 April 2024

1. Introduction

Due to their small size, low cost, and high maneuverability, drones are widely used in many applications, including target search, disaster rescue, electric power inspection, and so on. Among various capabilities of drones, autonomous path planning plays an important role in guaranteeing the task accomplishment of drones [1,2]. Despite extensive research on motion decision algorithms having been carried out, fast and accurate path planning in complex unknown environments remains challenging for drones.

Traditional path-planning algorithms mainly include RRT (rapidly exploring random tree), artificial potential field, A* algorithm, and so on [3–6]. Nguyen, T.H. et al., proposed a new path-planning method by inserting a universal pseudo-random number generator into RRT, which improved the convergence and effectiveness of path planning for the mobile robot [7]. Sabudin, E.N. et al., suggested a path potential field-based planning algorithm, which was capable of eliminating the local minima that frequently occurs in the conventional potential field while fulfilling the criterion of path planning and integrates path pruning to shorten the planned path [8]. To optimize path length and path smoothness, Zhang, L. et al., introduced an improved A* algorithm by dividing the distance between adjacent nodes and employing a cubic spline function to smooth the path [9]. For the intelligent unmanned system, the motion environment is not completely known. Certain adaptive and self-learning abilities are needed to adapt to changes in the environment and reach goals quickly. Although the above methods can solve the problems of the local minimum value of traditional path-planning algorithms to some extent, they are not suitable for fast path planning in the large-scale complex flight environments of drones.

With the development of AI (artificial intelligence), many AI-based path-planning algorithms have been conducted, providing inspiration for solving the problems of traditional motion decision-making algorithms. Machmudah, A. et al., addressed the optimization of flight trajectories for a fixed-wing UAV (unmanned aerial vehicle) at a constant altitude by employing a Bezier curve and meta-heuristic optimizations, including PSO (particle swarm optimization), which minimized the path length while satisfying the maximum curvature and collision avoidance constraints [10]. Hu, H. et al., proposed a new algorithm called APF-D3QNPER, combining the action output method of APF (artificial potential field) with the dueling double deep Q-network algorithm and introduced experience sample rewards in the experience playback portion of the traditional DRL (deep reinforcement learning) algorithm, overcoming the limitations of traditional path planning methods in unknown environments, such as reliance on high-precision maps, lack of generalization ability, and obstacle avoidance capability [11]. Xie, R. et al., presented a DRL approach for three-dimensional path planning utilizing local information and relative distance without global information to make up for the deficiencies of traditional path-planning algorithms in a complex and dynamic environment [12]. The above AI-based path-planning methods can effectively support the path-finding tasks of drones in three-dimensional complex environments but ignore dynamic perception in complex environments.

Animals have excellent navigation abilities. Drawing inspiration from animal environmental perception and navigation mechanisms, bio-inspired perception, navigation, and path-planning technologies provide a possibility to further solve intelligent path-planning issues in complex and unknown environments [13,14]. The neuromorphic visual system event camera is regarded as a potential candidate for improving the rapidity of perception and path planning for drones in dynamic environments. Compared with traditional machine vision systems, biology-inspired event cameras have significant advantages over traditional cameras: low response delay, low data rate, high dynamic range, and low power consumption, providing new ideas for improving the performance of existing AI-based path-planning algorithms. Although there are currently some studies on event-based reinforcement learning, these methods are mainly applied to fields like traffic signals, intelligent trains, multi-agent systems, etc. [15–17]. However, there is little research using the airsim unreal platform to simulate real dynamic characteristics and physical features, such as collisions, for the event-driven reinforcement learning training of unmanned aerial vehicles. In this paper, an intelligent perception and path-planning method of drones based on event camera and reinforcement learning named E-DQN is proposed to address the drawbacks of traditional algorithms and make up for the lack of application of event-driven reinforcement learning algorithms in the field of drone decision-making in airsim. Environment perception employing event data can effectively improve the rapidity of path-planning systems. Reinforcement learning draws inspiration from the dopamine reward and punishment mechanism in the brain and has strong environment adaptability and flexibility, which makes it a potential candidate for solving path-planning problems in large-scale and dynamically changing environments. Deep Q-network (DQN) is a commonly used optimization algorithm for reinforcement learning, which employs a neural network to represent the optimal policy function Q and updates the parameters. Reinforcement learning training by DQN can generate optimal value functions through interaction with the environment without prior knowledge [18]. Introducing event streams into DQN for path planning can ensure the stability and efficiency of training, overcoming the shortcomings of traditional decision algorithms. The proposed algorithm was validated on the airsim simulation platform [19]. The experiment results show that the optimal path to the target can be found after several rounds of training.

The remainder of this paper is structured as follows: Section 2 describes the overall framework of the proposed path-planning algorithm. The detailed method is explained in Section 3. Section 4 presents simulation experiments developed on a virtual flight platform supported by airsim and unreal engine. Section 5 concludes the whole work and discusses future research.

2. System Framework

Path planning is performed to generate a motion strategy with which a drone can find an optimal path connecting the starting and ending positions. Considering the complex task execution environment of drones, a bio-inspired path-planning algorithm based on event perception and reinforcement learning was designed to perform fast decision-making for drones. The overall framework is as Figure 1.

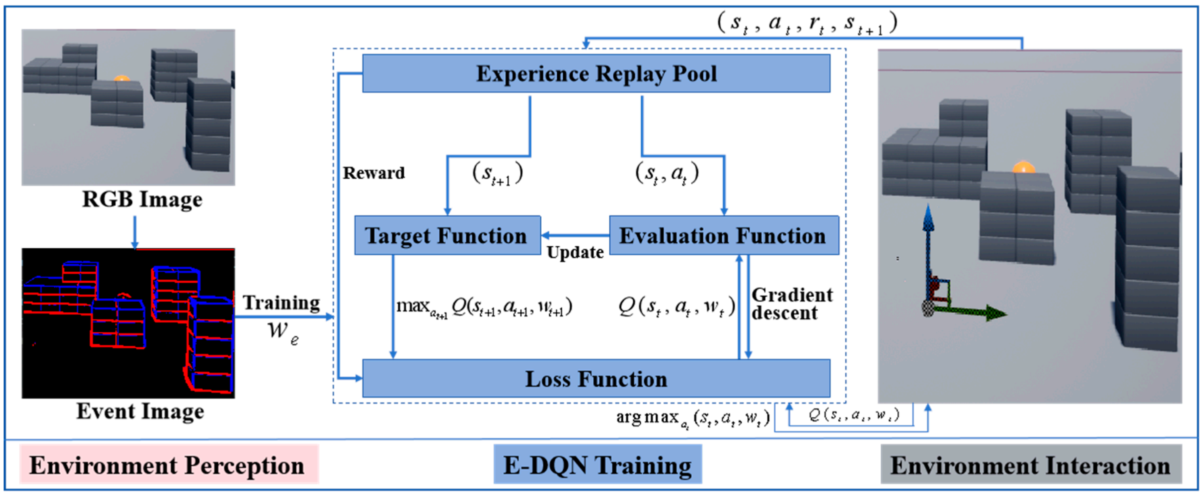

Figure 1. The overall framework of bio-inspired path planning system.

The system framework consisted of three parts, including environment perception, E-DQN training, and environment interaction. Event data were employed in the bio-inspired path-planning system for perception, which greatly reduced data redundancy and improves the training efficiency of DQN [20]. Due to the asynchrony and sparsity of event data, an auto-encoder was introduced to process the event stream data and make them available for the following training network [21]. The DQN algorithm requires multiple episodes of training through interactions between drones and environments. Airsim serves as a plugin for any virtual environment provided by the unreal engine and is regarded as an ideal platform for artificial intelligence training [22]. Unreal engine simulator creates a highly realistic virtual environment with high fidelity and avoids situations where drones are prone to crashes while flying in the real environment [23]. In addition, airsim can equip drones with multiple sensors and provide rich API interfaces to support the task execution of drones in different scenarios.

3. E-DQN-Based Path-Planning Method

3.1. Bio-Inspired Environment Perception

Drones equipped with RGB cameras have reaction times of tens of milliseconds, which is not enough for drones to perform fast navigation tasks in complex dynamic environments. An event-based camera is a novel visual sensor inspired by animal perception mechanisms. The time resolution of the event camera is in the microsecond range. Compared with traditional cameras that capture images at a fixed frame rate, event cameras have the advantage of low latency, high dynamic range, low power consumption, and high time resolution. Moreover, events are inherently generated by changes in brightness typically arising from motion, which makes event cameras natural complex environment perception sensors and a good fit for fast path planning with obstacle avoidance function. Therefore, event data were employed for environmental perception in this paper.

Airsim is a simulator built on an unreal engine that supports the realistic physical and visual simulation of drones or cars. Airsim provides an event camera data collection simulator that allows drones to fly and collect event data in a virtual environment, which is equivalent to equipping a drone with an event camera. To prevent damage to drones caused by flying in real environments, the airsim virtual simulator is utilized for event data collection. Event cameras generate "events" by measuring logarithmic brightness changes. A set of events contains four values, namely pixel position (x and y coordinates), timestamp, and polarity. An event has a polarity of +1 or −1 based on an increase or decrease in logarithmic brightness. Event data in airsim are obtained using a series of transformations of RGB images. An event occurs when the absolute change in logarithmic brightness exceeds a certain threshold. Event sequences are reported in the form of byte streams, which construct an event accumulation on a 2D frame known as an "event image". In event images, events with a polarity of +1 and −1 are visualized as red pixels and blue pixels, respectively.

To convert RGB images into event images, firstly, the logarithmic strength of the current frame was calculated [24].

$$L(u,t) = \log(0.299 I_R(u,t) + 0.587 I_G(u,t) + 0.114 I_B(u,t)), \tag{1}$$

where $L(u,t)$ means the logarithmic strength of the current frame. $I_R(u,t)$, $I_G(u,t)$, and $I_B(u,t)$ represent the brightness of RGB images in the red, green, and blue channels, respectively. Subsequently, all pixels are traversed, and the polarity of each pixel is calculated based on the threshold of the difference in logarithmic intensity between the current frame and the previous frame. The polarity of the current frame can be obtained as follows:

$$p(u,t) = \begin{cases} +1, \ if \ L(u,t) - L(u,t-1) > Th \\ -1, \ if \ L(u,t) - L(u,t-1) < -Th \end{cases}, \tag{2}$$

where Th denotes the event brightness threshold. According to the degree to which the intensity change exceeds the threshold, the number of events to be emitted for each pixel can be determined. Assuming that the maximum number of events that a pixel experience is $N_{\max}$, the total number of emissions simulated at the pixel position u can be expressed as follows:

$$N_{\max} = \text{int}\left(\frac{\Delta t}{r_t \times 10^{-3}}\right), \tag{3}$$

$$N_e(u,t) = \min\left(N_{\max}, \frac{\Delta L(u,t)}{TOL}\right), \tag{4}$$

where r_t is the refractory period. $N_e(u,t)$ represents the number of events generated at pixel u. $N_{\max}$ represents the maximum constraint on the maximum number of possible events that can occur in a pixel. $\Delta L(u,t)$ is the total logarithmic brightness change that occurs at pixel u and time t.

To train the neural network architecture using event data, a one-dimensional event stream with event data form (x, y, t, p) was generated, where (x, y) denotes the sum coordinates of the current pixel, t is the timestamp, and p represents the polarity of the event. Then, the timestamp of each interpolation event was determined through interpolation, which represents the amount of time between the captured previous and current images.

$$t = t_{prev} + \frac{\Delta T}{N_e(u)}, \tag{5}$$

where t_{prev} means the timestamp of the first event generated at the current time step. ΔT indicates the time interval between frames in microseconds. Finally, the generated event stream was sorted in timestamp order to simulate event data correctly.

3.2. Feature Extraction of Event Data

The generated event data cannot be applied to neural network-based motion decision algorithms directly because these data are not images but asynchronous event streams encoding changes in pixel intensity. The data processing of an asynchronous event stream is one of the key issues in utilizing event data for bio-inspired navigation and decision-making systems.

With the resolution of the event camera set to (H, W), the event at time t was defined as a tuple (t, x, y, p). A series of events in the time window τ were represented as $E_\tau = \{e_i | t < i < t + \tau\}$. The events in E_τ can be accumulated and represented as the corresponding event image frames IE_τ. Although event sequences are time-based data streams, the data length is long and difficult to apply to neural networks. Therefore, the decoupling of temporal and spatial information is needed to apply event data to path-planning and obstacle-avoidance algorithms. For spatial representation, pixel coordinates and polarity were encoded separately by calculating and extracting asynchronous event stream features. Moreover, the principle of position encoding was utilized for time embedding. For an event set E_n with n events, the timestamp was normalized between 0 and 1 so that the timestamp corresponding to the end of the window was mapped to 1. Then, time characteristics were calculated for each normalized timestamp. The spatiotemporal decoupling process of event data is shown in Figure 2 [25].

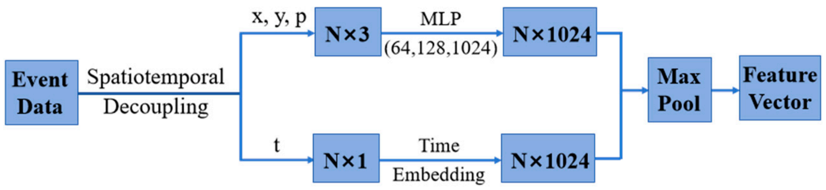

Figure 2. Spatiotemporal decoupling of event data.

An auto-encoder is used to restore input data at the output and extract useful information from the data, which acts as a bridge for applying event data to reinforcement learning training. The auto-encoder describes a probabilistic framework for mapping feature vectors to reconstructed space rather than randomly mapping attributes to the output [26]. The auto-encoder attempts to learn parameter variable models by maximizing the marginal log-likelihood of training data composed of reconstruction loss and KL divergence loss, as shown in Figure 3.

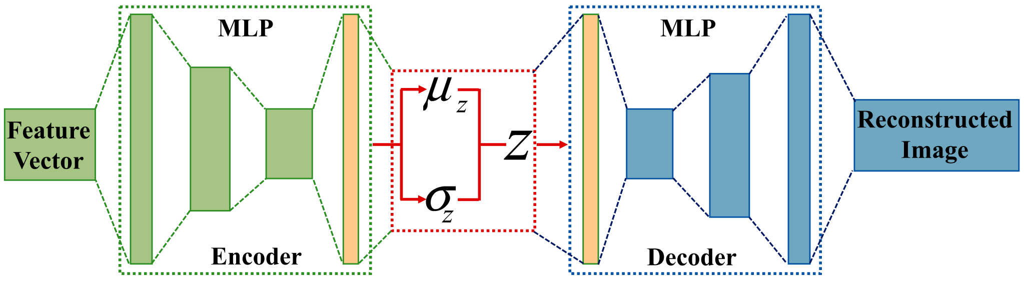

Figure 3. Auto-encoder of feature vector.

Due to the narrow bottleneck layer in the middle of the network, only the most important information is preserved in the encoding value z. The decoder inputs a vector z and outputs a reconstructed image. Multi-layer perceptron (MLP) was used as a classification model for the encoder and the decoder. The activation function was implemented to solve the classification model of MLP. The commonly used activation functions include the sigmoid function, the tanh function, and the ReLU function. The sigmoid function and tanh function have a gradient saturation and output of non-zero mean at both ends of the function, which can affect the training effectiveness of the neural network. The ReLU function returns the input value when it is greater than 0 and 0 when it is less than or equal to 0 [27].

$$\mathrm{ReLU}(x) = \max(0, x) \tag{6}$$

The ReLU function is simple and efficient in calculation without involving complex exponential operations. Moreover, the ReLU function can solve the gradient saturation problem of the sigmoid and tanh functions. Furthermore, the derivative of the activation function is always constant 1 in the positive region, which can effectively transfer gradients [28]. Therefore, the ReLU function was adopted as the activation function here. The training of the auto-encoder was performed end-to-end. The weights of the auto-encoder were generated simultaneously. The trained weights w_e of the auto-encoder could be utilized as inputs for path-planning algorithms.

3.3. E-DQN Training of Drones

The path-planning problem refers to finding a collision-free path under the constraints of path length and training time. Reinforcement learning is a learning method inspired by animal dopamine reward strategies, suitable for solving path-planning problems through continuous trial and error. Animals produce movements through the prefrontal cortex. Dopamine is a neurotransmitter produced by the brain that can provide feedback on reward signals based on actions and states, thereby generating behaviors such as homing and foraging. The decision-making mechanism of animals is shown in Figure 4.

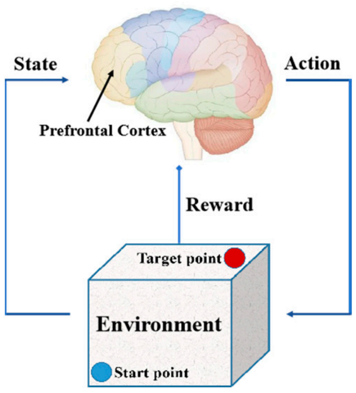

Figure 4. The decision-making mechanism of animals.

Inspired by an animal's path-planning mechanism, reinforcement learning aims to learn motion policies for sequential decision problems by optimizing a cumulative future-reward signal. Reinforcement learning has exploratory characteristics and can continuously try new actions to discover reward signals during the learning process, which makes reinforcement learning highly adaptable and flexible when facing unknown environments or new tasks. Through exploration, drones can gradually learn the characteristics and laws of the environment, thereby generating optimal motion strategies. With event data representation value z as state observations, generating the next action by reinforcement learning can effectively solve the problem of the fast path planning of drones in complex environments.

Q-learning is one of the most popular reinforcement learning methods. However, in many cases, the state space of reinforcement learning tasks is continuous and infinite, making it difficult to store the value function in a table format [29]. The DQN algorithm can improve the processing ability of high-dimensional state spaces, providing a new idea to address the issues of rapidity and continuity in decision-making.

DQN approximates the action value $Q(s, a)$ by adopting a value function $Q(s, a; w)$, where w is the parameter for training the neural network [30]. Firstly, the original state s_t and Q values of corresponding actions should be initialized. By providing starting position $O(O_x, O_y, O_z)$, target position $T(T_x, T_y, T_z)$ and event weights W_e, the next action selection of drones can be performed. In the decision-making process, $\varepsilon - greedy$ strategy, which means a drone select actions randomly with a certain probability ε, is adopted to select an action and react with the environment to obtain a new state s_{t+1} and reward r. The $\varepsilon - greedy$ strategy helps to balance the trade-off between exploration and utilization, preventing getting stuck in a local optimum.

The training of path-planning neural networks is an optimization problem. The DQN algorithm includes two neural networks: the evaluated network (*Q*-value network) and the target network. The target network is to train the network by minimizing the mean square error of the Q value so that the network can approximate the real Q function. The objective function includes the difference between the actual reward and the target Q value of the next state. The calculation formula of the target network can be written as follows [31]:

$$y_t = r + \gamma \cdot \max_a Q(s_{t+1}, a; w, w_e), \tag{7}$$

where y_t denotes the Q value of the target network, which represents the expected reward obtained after performing an action. γ describes the discount factor coefficient. s_{t+1} denotes the next state. a is the selected action. w refers to the weights of the target network at time t. w_e means event weights. $Q(s_{t+1}, a; w, w_e)$ represents the maximum Q value corresponding to the action in state s_{t+1}. r stands for the reward and can be written as follows:

$$r = \begin{cases} 100, \text{ Reach the goal} \\ -100, \text{ If collosion} \\ 0, \text{ Others} \end{cases} \tag{8}$$

To optimize the training network, the difference between the evaluated network and the target network is calculated as the following loss function [32]:

$$L = 1/2[y_t - Q(s, a; w, w_e)]^2, \tag{9}$$

where L is the loss function. Then, the Q parameters of the neural network are updated using the back propagation gradient descent method to make the $Q(s_t, a_t)$ approach y_t. The updated Q network is the following [33]:

$$Q = r + (\gamma * Q_{t+1} * (1 - dones)), \tag{10}$$

where *dones* equals 0 or 1 indicates whether to complete a training session.

A transition of a quadruple (s_t, a_t, r_t, s_{t+1}) can be obtained through the update of the Q value. Then, a new state is entered to repeat the above process. After multiple training processes, the optimal path in the obstacle environment can be found. There is a strong correlation between adjacent transitions, which have an impact on Q-network training if transitions are used in order. Therefore, experience replay is introduced to eliminate correlation, provide convergence speed, and improve data utilization. A replay buffer that stores n transitions, known as experience, will be created. During each episode of training, batch-size transition data are selected randomly, and multiple random gradients are calculated to update the parameter w of the target Q-network. The flowchart of the proposed E-DQN training process for the bio-inspired path planning of drones in complex environments is shown in Algorithm 1.

Algorithm 1. E-DQN training process for bio-inspired path planning of drones in complex environments.

```
1: Input: Starting point $O(O_x, O_y, O_z)$; Target point $T(T_x, T_y, T_z)$; Event weight $W_e$
2: Output: Optimal path
3: Initialize: Evaluation function; Target function; State $s_t$
4: **for** $t < t_{end}$
5:    Select an action $a_t$ by the $\varepsilon - greedy$ strategy.
6:    Input $a_t$ into the environment and obtain new states $s_t$ and $r$.
7:       Compute target function $y_t = r_t + \gamma \cdot \max_a Q(s_{t+1}, a; w)$
8:       Compute loss function $L = 1/2[y_t - Q(s, a; w)]^2$
9:       Introduce experience replay and update parameters
         $Q = r + (\gamma * Q_{t+1} * (1 - dones))$
10: **end for**
```

4. Experiments

Plan planning with obstacle avoidance means that drones can independently analyze the environment information and generate a collision-free path from the initial state to the target state in an environment with unknown obstacles. To verify the feasibility of the proposed algorithm, an obstacle-avoidance scenario built on the airsim platform was created.

4.1. Experimental Setup

The simulation was performed on a computer with a 2.40 GHz Intel Core i7 processor and an 8 GB GPU. To interact with airsim's APIs, programming was implemented using the Python language. The flight range of the drone was a three-dimensional obstacle environment of 110 m $\times$ 110 m $\times$ 25 m, as shown in Figure 5.

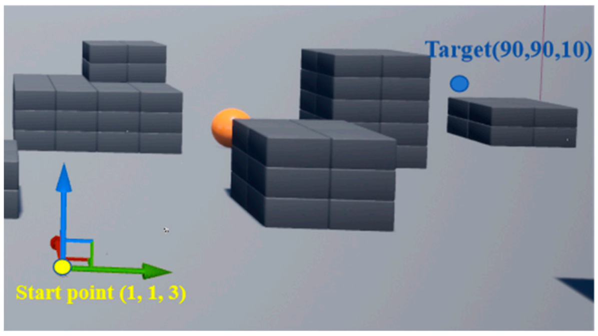

Figure 5. Flight environment.

In the airsim flight environment, the positions of the yellow dot and the blue dot were regarded as the starting point and the target point, respectively. The gray blocks and yellow ball were considered obstacles. The start position and the target position were set to (1, 1, 3) and (90, 90, 10), respectively. The goal of the experiment was to enable a drone to find an optimal and collision-free path from the start to the target. The path-planning method was trained using E-DQN. To ensure that the drone is well-trained, the total training timesteps were set to 5×10^5. The main parameters utilized in the experiment are shown in Table 1.

Table 1. Parameter settings of DQN.

Parameter	Value
Discount rate	0.99
Time constant	0.005
Batch_size	128
Hidden_dim	16
Learning_rate	0.0004
Num_episodes	40,000
Max_eps_episode	10
threshold	200
State_dim	42
Action_dim	27

4.2. Experimental Results

To collect environmental information, a drone is allowed to fly freely in the airsim environment to collect RGB image data and generate event data by calculating the brightness changes in adjacent RGB images. The comparison of the RGB image and the event image is shown in Figure 6.

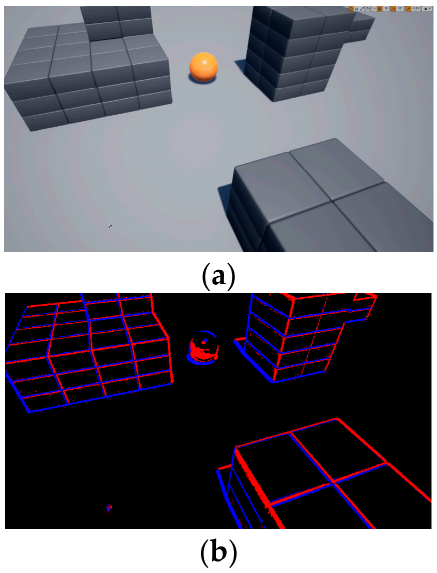

Figure 6. Comparison of RGB image and event image. (**a**) RGB image. (**b**) Event image.

Then, event weights generated by the auto-encoder were employed for DQN training and to obtain motion selection experience at a faster pace. Due to the lack of experience in the early stages of training, the drone randomly selected actions to explore the environment. The random exploration period was set as 1000. Moreover, the $\varepsilon - greedy$ strategy was adopted to select the next action of drones during one training cycle. When the drone entered a termination state due to a crash, reaching the target point, or exceeding the maximum step size, it entered the next training cycle. After training, the drone was able to reach the target point safely, and the motion trajectory of the drone could be obtained. To have a clearer view of the drone's flight situation, a proportional sketch of the airsim environment and the final path are shown in Figure 7.

It can be seen from Figure 7 that the drone can find a collision-free and shortest path to reach the target after training using the proposed method. The running time of the entire planning process was 41.364 s. Moreover, the entire planning process consisted of 188 steps.

Figure 8 shows the reward values for different episodes. The total episodes was 16,381 during the training process. In Figure 8, the red dots represent the highest reward value of 195, while other reward values are represented by blue dots. It can be seen from Figure 8 that the training process converges gradually toward 6000 episodes, and most of the reward values can receive the maximum reward value during 6000–16,381 episodes, which proves the convergence speed of the proposed algorithm.

To further demonstrate the performance of the proposed method, the running time of the E-DQN path-planning method was compared with that of DQN and the spiking neural network (SNN) [31] in the same test environment. DQN is a commonly used intelligent path-planning algorithm, and SNN is also one of the typical brain-inspired path-planning methods. In our previous research, the running time of the SNN path-planning algorithm was introduced in detail [34]. The path-planning algorithm utilizing SNN adopts the leaky integrate-and-fire (LIF) model to generate a motion strategy. The parameter settings for SNN are shown in Table 2. The comparison results are shown in Table 3.

The path-planning time using the E-DQN algorithm was significantly reduced compared with that of the DQN algorithm and SNN algorithm, which verifies the rapidity and feasibility of the proposed algorithm.

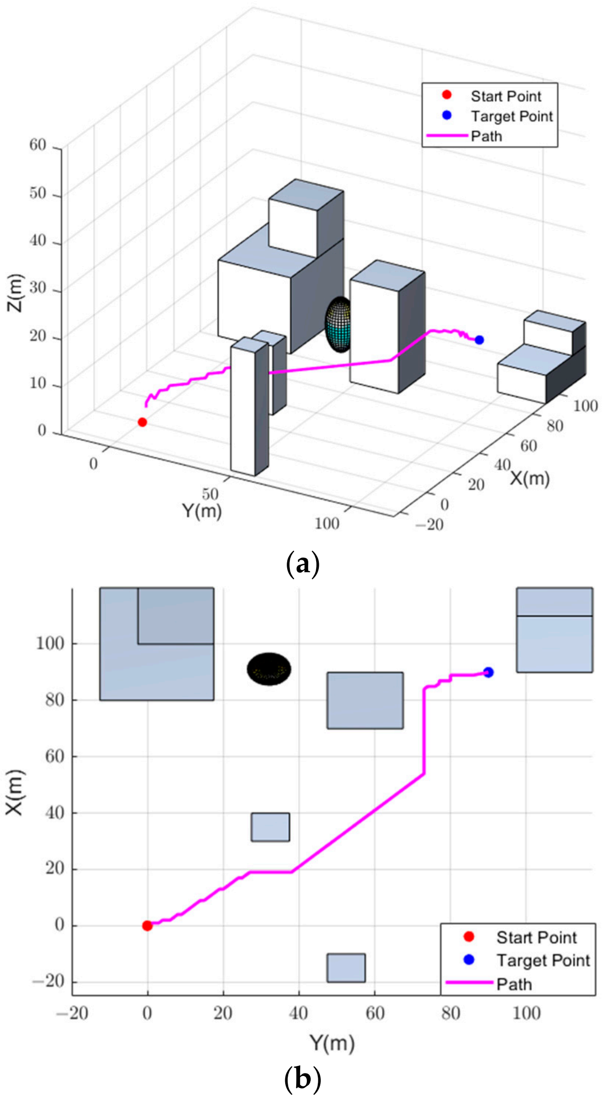

Figure 7. Planned path. (**a**) 3D View of the Planed Path. (**b**) Top View of the Planned Path.

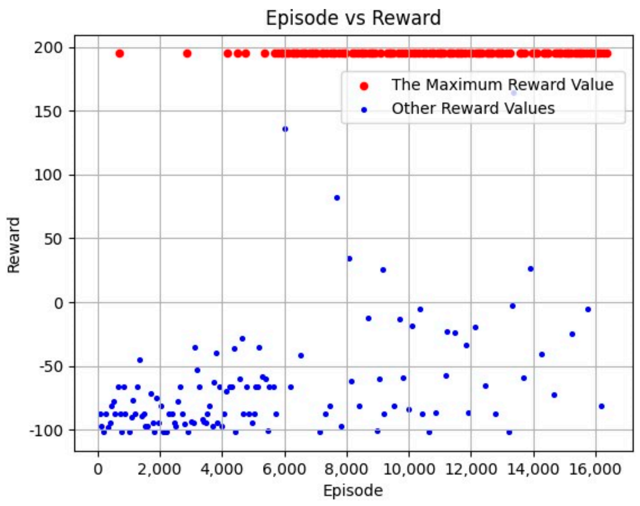

Figure 8. Episodes vs. Reward Result.

Table 2. The parameter settings for SNN.

Parameter	Value
membrane time constant	0.002
rest membrane potential	0 mV
membrane resistance	20 M
pre-amplitude	1.0
post-amplitude	1.0
pre-time constant	0.0168
post-time constant	0.0337

Table 3. Runtime comparison of typical brain-inspired path-planning methods.

Method	Run Time (s)
E-DQN	41.364
DQN	161.88
SNN	155.4

5. Conclusions

To improve the rapidity of motion decision-making, a bio-inspired path-planning algorithm based on E-DQN for drones in complex environments was proposed. Unreal engine and airsim plugin were employed to build a training environment with obstacles for drones. Event data were generated using airsim simulator environment perception to reduce information redundancy. In terms of the asynchrony and sparsity of event data, an auto-encoder was designed to generate event weights, which were then input into a reinforcement learning network for training. Subsequently, DQN was introduced for the motion planning training of drones, and the ε-greedy strategy was used for drone action selection. The next step, the state of the drone, could be updated by setting reward signals and interacting with the airsim simulation environment. After several training sessions, the drone could find an optimal collision-free path to the target. The key contributions of this paper are listed below.

(1) An event auto-encoder for the unsupervised representation learning is presented from fast and asynchronous spatiotemporal event byte stream data.
(2) Motion policies are trained over event representations using the DQN for path planning for drones, which is superior to traditional path-planning algorithms.
(3) The dopamine reward mechanism is adopted for obstacle avoidance, which is more in line with animal action decision-making behavior.

Future work will investigate the extension of our algorithm to multiple drones and dynamically growing maps for motion decision-making and conduct real-world field experiments to validate our method. However, the real-world application of the proposed method also has limitations and challenges, such as efficiency issues, security issues, interpretability issues, and the practical challenges and scalability of the proposed method in diverse real-world scenarios. Overall, the problems of the E-DQN algorithm in practical applications are diverse and complex. However, with the continuous deepening of research and the continuous development of technology, many problems are expected to be solved or alleviated.

Author Contributions: Methodology, Y.C.; software, A.R.; writing—original draft preparation, R.D.; writing—review and editing, Z.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by [Science and Technology Bureau] grant number [JCKY2020605C009] and [Aeronautic Science Foundation of China] grant number [ASFC-2020Z071052001].

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to the research project.

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

References

1. Ramalepa, L.P.; Jamisola, R.S. A review on cooperative robotic arms with mobile or drones bases. *Int. J. Autom. Comput.* **2021**, *18*, 536–555. [CrossRef]
2. Quinones, G.M.; Biswas, G.; Ahmed, I.; Darrah, T.; Kulkarni, C. Online decision making and path planning framework for safe operation of unmanned aerial vehicles in urban scenarios. *Int. J. Progn. Health Manag.* **2021**, *12*, 1–17.
3. Martinez, N.A.; Lu, L.; Campoy, P. Fast RRT* 3D-sliced planner for autonomous exploration using MAVs. *Unmanned Syst.* **2022**, *10*, 175–186. [CrossRef]
4. Mohammed, H.; Romdhane, L.; Jaradat, M.A. RRT* N: An efficient approach to path planning in 3D for static and dynamic environments. *Adv. Robot.* **2021**, *35*, 168–180. [CrossRef]
5. Rostami, S.M.H.; Sangaiah, A.K.; Wang, J.; Liu, X. Obstacle avoidance of mobile robots using modified artificial potential field algorithm. *EURASIP J. Wirel. Commun. Netw.* **2019**, *1*, 70. [CrossRef]
6. Duchon, F.; Babinec, A.; Kajan, M.; Beňo, P.; Florek, M.; Fico, T.; Jurišica, L. Path planning with modified a star algorithm for a mobile robot. *Procedia Eng.* **2014**, *96*, 59–69. [CrossRef]
7. Nguyen, T.H.; Nguyen, X.T.; Pham, D.A.; Tran, B.L.; Bui, D.B. A new approach for mobile robot path planning based on RRT algorithm. *Mod. Phys. Lett. B* **2023**, *37*, 2340027. [CrossRef]
8. Sabudin, E.N.; Omar, R.; Debnath, S.K.; Sulong, M.S. Efficient robotic path planning algorithm based on artificial potential field. *Int. J. Electr. Comput. Eng.* **2021**, *11*, 4840–4849. [CrossRef]
9. Zhang, L.; Li, Y. Mobile robot path planning algorithm based on improved a star. *J. Phys. Conf. Ser.* **2021**, *1848*, 012013. [CrossRef]
10. Machmudah, A.; Shanmugavel, M.; Parman, S.; Manan, T.S.A.; Dutykh, D.; Beddu, S.; Rajabi, A. Flight trajectories optimization of fixed-wing UAV by bank-turn mechanism. *Drones* **2022**, *6*, 69. [CrossRef]
11. Hu, H.; Wang, Y.; Tong, W.; Zhao, J.; Gu, Y. Path planning for autonomous vehicles in unknown dynamic environment based on deep reinforcement learning. *Appl. Sci.* **2023**, *13*, 10056. [CrossRef]
12. Xie, R.; Meng, Z.; Wang, L.; Li, H.; Wang, K.; Wu, Z. Unmanned aerial vehicle path planning algorithm based on deep reinforcement learning in large-scale and dynamic environments. *IEEE Access* **2021**, *9*, 24884–24900. [CrossRef]
13. Banino, A.; Barry, C.; Uria, B.; Blundell, C.; Lillicrap, T.; Mirowski, P.; Pritzel, A.; Chadwick, M.J.; Degris, T.; Modayil, J.; et al. Vector-based navigation using grid-like representations in artificial agents. *Nature* **2018**, *557*, 429–433. [CrossRef] [PubMed]
14. Edvardsen, V. Goal-directed navigation based on path integration and decoding of grid cells in an artificial neural network. *Nat. Comput.* **2019**, *18*, 13–27. [CrossRef]
15. Wang, S.; Xie, X.; Huang, K.; Zeng, J.; Cai, Z. Deep reinforcement learning-based traffic signal control using high-resolution event-based data. *Entropy* **2019**, *21*, 744. [CrossRef] [PubMed]
16. Zhang, L.; Zhou, M.; Li, Z. An intelligent train operation method based on event-driven deep reinforcement learning. *IEEE Trans. Ind. Inform.* **2021**, *18*, 6973–6980. [CrossRef]
17. Menda, K.; Chen, Y.C.; Grana, J.; Bono, J.W.; Tracey, B.D.; Kochenderfer, M.J.; Wolpert, D. Deep reinforcement learning for event-driven multi-agent decision processes. *IEEE Trans. Intell. Transp. Syst.* **2018**, *20*, 1259–1268. [CrossRef]
18. Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Horgan, D.; Quan, J.; Sendonaris, A.; Osband, I.; et al. Deep q-learning from demonstrations. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 29 April 2018; Volume 32.
19. Shah, S.; Dey, D.; Lovett, C.; Kapoor, A. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics: Results of the 11th International Conference*; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 621–635.
20. Falanga, D.; Kleber, K.; Scaramuzza, D. Dynamic obstacle avoidance for quadrotors with event cameras. *Sci. Robot.* **2020**, *5*, 9712. [CrossRef] [PubMed]
21. Andersen, P.A.; Goodwin, M.; Granmo, O.C. Towards safe reinforcement-learning in industrial grid-warehousing. *Inf. Sci.* **2020**, *537*, 467–484. [CrossRef]
22. Xie, D.; Hu, R.; Wang, C.; Zhu, C.; Xu, H.; Li, Q. A simulation framework of unmanned aerial vehicles route planning design and validation for landslide monitoring. *Remote Sens.* **2023**, *15*, 5758. [CrossRef]
23. Buck, A.; Camaioni, R.; Alvey, B.; Anderson, D.T.; Keller, J.M.; Luke, R.H., III; Scott, G. Unreal engine-based photorealistic aerial data generation and unit testing of artificial intelligence algorithms. In *Geospatial Informatics XII*; SPIE: Orlando, FL, USA, 2022; Volume 12099, pp. 59–73.
24. Available online: https://microsoft.github.io/AirSim/event_sim/ (accessed on 30 December 2023).
25. Vemprala, S.; Mian, S.; Kapoor, A. Representation learning for event-based visuomotor policies. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 4712–4724.
26. Zhou, J.; Komuro, T. Recognizing fall actions from videos using reconstruction error of variational autoencoder. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 3372–3376.

27. Si, J.; Harris, S.L.; Yfantis, E. A dynamic ReLU on neural network. In Proceedings of the 2018 IEEE 13th Dallas Circuits and Systems Conference (DCAS), Dallas, TX, USA, 12 November 2018; pp. 1–6.
28. Rynkiewicz, J. Asymptotic statistics for multilayer perceptron with ReLU hidden units. *Neurocomputing* **2019**, *342*, 16–23. [CrossRef]
29. Low, E.S.; Ong, P.; Cheah, K.C. Solving the optimal path planning of a mobile robot using improved Q-learning. *Robot. Auton. Syst.* **2019**, *115*, 143–161. [CrossRef]
30. Coy, M.V.C.; Casallas, E.C. Training neural networks using reinforcement learning to reactive path planning. *J. Appl. Eng. Sci.* **2021**, *19*, 48–56.
31. Ab Azar, N.; Shahmansoorian, A.; Davoudi, M. Uncertainty-aware path planning using reinforcement learning and deep learning methods. *Comput. Knowl. Eng.* **2020**, *3*, 25–37.
32. Kim, C. Deep Q-Learning Network with Bayesian-Based Supervised Expert Learning. *Symmetry* **2022**, *14*, 2134. [CrossRef]
33. Rubí, B.; Morcego, B.; Pérez, R. Quadrotor path following and reactive obstacle avoidance with deep reinforcement learning. *J. Intell. Robot. Syst.* **2021**, *103*, 62. [CrossRef]
34. Chao, Y.; Augenstein, P.; Roennau, A.; Dillmann, R.; Xiong, Z. Brain inspired path planning algorithms for drones. *Front. Neurorobotics* **2023**, *17*, 1111861. [CrossRef] [PubMed]