drones

MDPI

Article

Efficient Focus Autoencoders for Fast Autonomous Flight in Intricate Wild Scenarios

Kaiyu Hu [1,†], Huanlin Li [1,†], Jiafan Zhuang [1,2,3,*], Zhifeng Hao [1,*] and Zhun Fan [1,2,3,*]

[1] College of Engineering, Shantou University, Shantou 515063, China; 22kyhu@stu.edu.cn (K.H.); 21hlli@stu.edu.cn (H.L.)
[2] Key Laboratory of Intelligent Manufacturing Technology, Shantou University, Ministry of Education, Shantou 515063, China
[3] International Cooperation Base of Evolutionary Intelligence and Robotics, Shantou University, Shantou 515063, China
[*] Correspondence: jfzhuang@stu.edu.cn (J.Z.); haozhifeng@stu.edu.cn (Z.H.); zfan@stu.edu.cn (Z.F.)
[†] These authors contributed equally to this work.

Abstract: The autonomous navigation of aerial robots in unknown and complex outdoor environments is a challenging problem that typically requires planners to generate collision-free trajectories based on human expert rules for fast navigation. Presently, aerial robots suffer from high latency in acquiring environmental information, which limits the control strategies that the vehicle can implement. In this study, we proposed the SAC_FAE algorithm for high-speed navigation in complex environments using deep reinforcement learning (DRL) policies. Our approach consisted of a soft actor–critic (SAC) algorithm and a focus autoencoder (FAE). Our end-to-end DRL navigation policy enabled a flying robot to efficiently accomplish navigation tasks without prior map information by relying solely on the front-end depth frames and its own pose information. The proposed algorithm outperformed existing trajectory-based optimization approaches at flight speeds exceeding 3 m/s in multiple testing environments, which demonstrates its effectiveness and efficiency.

Keywords: deep reinforcement learning; soft actor–critic; focus autoencoder; unmanned aerial vehicle; autonomous navigation

check for updates

Citation: Hu, K.; Li, H.; Zhuang, J.; Hao, Z.; Fan, Z. Efficient Focus Autoencoders for Fast Autonomous Flight in Intricate Wild Scenarios. *Drones* **2023**, *7*, 609. https://doi.org/10.3390/drones7100609

Academic Editors: Enrico Boni and Michele Basso

Received: 12 August 2023
Revised: 18 September 2023
Accepted: 22 September 2023
Published: 27 September 2023

1. Introduction

Flying robots are among the most flexible man-made robots developed to date. With their high level of maneuverability, these robots can navigate through complex and challenging environments, including natural forests and modern urban buildings, and they can reach areas that most other human-made robots cannot. Their flexibility has led to the development of numerous applications, including environmental mapping [1,2], patrol inspection [3], search and rescue [4], logistics automation [5], entertainment performances [6], and agricultural automation [7]. Flying robots have an incredible level of autonomy, thereby enabling them to achieve previously impossible missions. To this end, the autonomous navigation capability of a flying robot enables it to safely interact with the environment and fly to its destination automatically without human intervention.

The autonomous navigation capability of the flying robot enables the aircraft to safely interact with the environment and fly to its destination automatically without human intervention. After years of continuous development and research [8,9], many previous research efforts have led to the development of flying robots in the field of full autonomy, thus freeing the hands of human expert pilots. However, developing end-to-end navigation methods that are capable of robust flight at high speeds in complex environments is a long-standing challenge that remains unsolved. Traditional trajectory-based optimization methods need to prebuild a mathematical model for the environment, which usually requires a map construction procedure, such as the ESDF map [10–16]. However, the mapping procedure tends to be time-consuming, and it is difficult to meet the real-time requirements. Imitation-learning-based methods train a policy generator based on a large amount of expert experiences [17,18]. However, by only imitating the human experience, the learned policy generator cannot handle unseen scenarios and would make a inappropriate decision.

In recent years, learning-based end-to-end algorithms such as soft actor–critic (SAC) [19] combined with convolutional neural networks have been investigated. SAC is an offline deep reinforcement learning (DRL) algorithm that employs a stochastic policy to maximize the sample utilization. These end-to-end algorithms can directly map the visual observations, attitude, and desired target information of a flying robot to the action output of the agent. In related work [20], a proposed sensor-level DRL-based policy surpassed traditional algorithms in complex pedestrian scenario navigation tasks using a ground robot platform, which was greatly impressive. Although neural-network-based methods may be less explainable, they are still preferred, since they do not require rigorous mathematical proofs or tedious theoretical analyses. In the study of Xue et al. [21], seven ranging sensors were used for the perception of the environment, and a reinforcement learning approach based on an actor–critic framework was used to the achieve autonomous navigation of the UAV in an unknown environment. Similarly, in Zhang et al.'s study [22], more than seven laser ranging sensors were used to sense the environment, and an improved TD3-based algorithm was used to realize an autonomous navigation task for a UAV in a multi-obstacle environment. However, both methods are dependent on ranging sensors, and neither of them can accurately perceive the environment in the UAV's forward direction.

In this work, our algorithm assumption is that a UAV can achieve intelligent navigation by only relying on visual inputs and its own attitude information. The reason behind this assumption is inspired by human experiences. Humans can achieve autonomous exploration and navigation in an unknown and complex environments based on visual observation. Therefore, we follow the deep reinforcement learning route and focus on designing an efficient visual processing procedure. Our proposed method mainly utilizes vision sensors and onboard inertial measurements to achieve the robust autonomous navigation of flying robots. In contrast to existing works, it does not require computation or prebuilt maps, which can effectively reduce the perception latency. To tackle the low-sample complexity issue [23] in DRL, a previous work, SAC_RAE [24], has been proposed to use a regularized autoencoder (RAE) to project raw pixels into a latent state before being fed into the DRL module. However, to improve the processing efficiency, the SAC_RAE downsamples the projected feature map in the autoencoder multiple times, which results in significant information loss. In this work, we propose a focus autoencoder (FAE) to enhance the representative ability of the features with a large receptive field while still keeping the processing procedure efficient. We conducted extensive experiments to validate the effectiveness of our method, as shown in Figure 1. Our proposed SAC_FAE can outperform SAC_RAE and trajectory-based optimization methods by 10% and 19% in the navigation success rate, respectively. In addition, our proposed SAC_FAE outperformed other methods in multiple test environments, which validates the effectiveness and good robustness of our method. Our code implementation is available at https://github.com/LHL6666/SAC_FAE (accessed on 23 September 2023).

The rest of this paper is organized as follows. We review the related works on the autonomous navigation of aerial robots in Section 2. Section 3 provide the details of our proposed method. Section 4 and Section 5 provide a detailed introduction of our experimental setup and an evaluation of the experimental results, respectively. Finally, we conclude the work in Section 6.

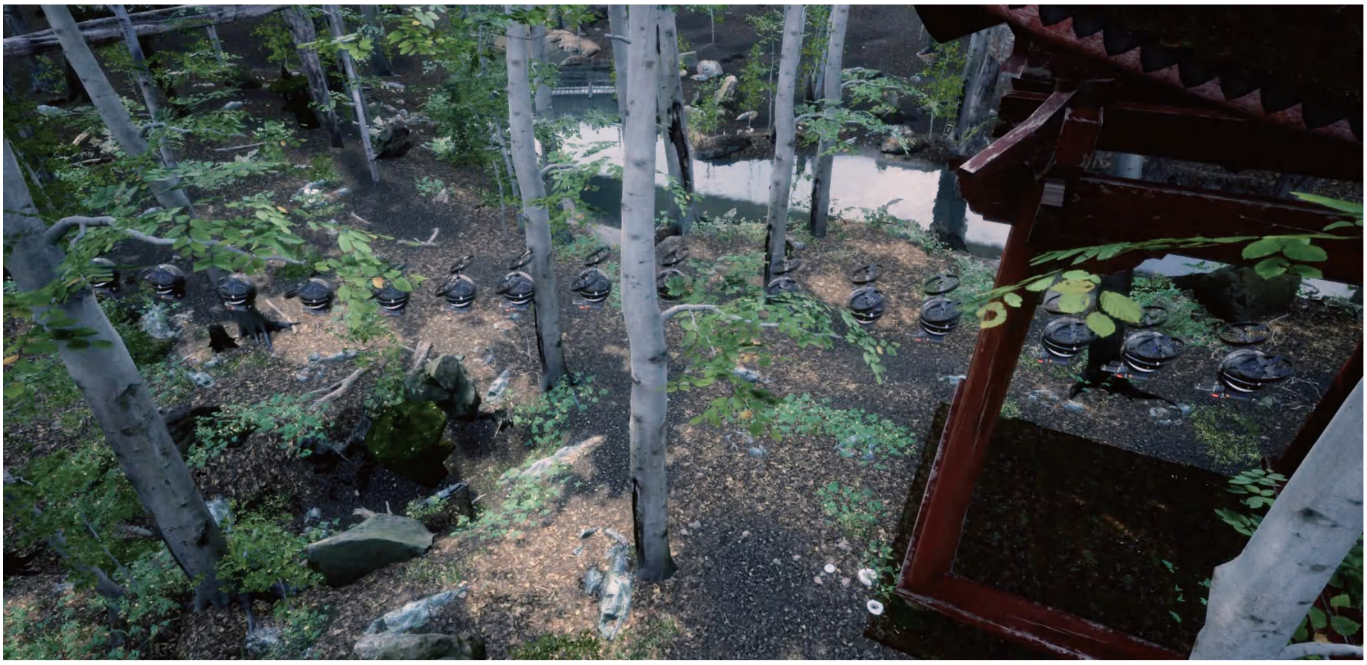

Figure 1. A snapshot of a flying robot deployed with the proposed method flying in one of the experimental scenarios. Trajectories shown of proposed flying robot autonomously navigating through a natural forest scene.

2. Related Works

In the field of automatic navigation for flying robots, various approaches to achieve autonomous flight have been proposed in the literature, which can be broadly categorized into the following three types: (1) Trajectory-based optimization methods: These methods involve designing a set of optimal trajectories that the robot should follow to reach its destination. They commonly rely on mathematical models and algorithms to generate the trajectories, and they typically require accurate information about the environment and the robot's dynamics; (2) Imitation-learning-based methods: These methods require a large amount of expert experience to fit an AI model that performs well in specific environments, but they have poorer generalization and exploration capabilities; (3) Reinforcement-learning-based methods: These methods represent a promising approach to achieving autonomous flight, which involves training an intelligent agent to learn how to navigate by interacting with its environment and receiving feedback in the form of rewards or penalties. Reinforcement-learning-based methods require a large amount of data for training, but they can use simulation software to obtain this data, thus making them more cost-effective than other methods. Additionally, the agent can continuously explore and learn from its environment, thus ultimately achieving comparable results to human experts. Next, we will introduce these methods in more detail.

2.1. Trajectory-Based Optimization Algorithms

Fast-Planner [25] and EGO-Planner [26] utilize certain search rules to find collision-free paths and optimize those paths for dynamic feasibility and smoothness. Fast-Planner features its stability, which is based on the approach of projecting depth images onto point clouds to construct ESDF maps and subsequently performing a path search and trajectory optimization. Since the planning algorithm needs to operate on the constructed ESDF map, the delay of the observation information becomes more prominent. This also means that, in order to achieve better performance, the speed of the flying robot must be strictly limited. Moreover, it should be noted that, due to the adaptive modification of the target point by trajectory optimization, Fast-Planner is not suitable for tasks in challenging environments requiring high-precision navigation. In navigation experiments conducted in complex scenes, the planner may exhibit conservative behaviors, because the target point does not impose a sufficient constraint on the behaviors, thereby resulting in a higher likelihood of task timeout without completion.

EGO-Planner is an improved planning algorithm that is based on Fast-Planner with an improved decision-making ability. This reduces the probability of task timeout while increasing the success rate. Interestingly, even when the planning horizon of EGO-Planner is increased several times, the algorithm still boldly explores and plans trajectories filled with exciting maneuvers such as frequent emergency turns for flying robots. In addition, the planner requires frequent restarts to reduce data errors.

For navigation in complex unknown environments, these typical algorithms combine online mapping and traditional planning algorithms. From an engineering perspective, splitting the navigation task into environmental perception and local planning is attractive, because each component can be performed in parallel, thereby making the overall system more efficient and interpretable. However, there is a time–space mismatch between the output of the perception module and the joint debugging of the planner, which makes the interaction between different stages amenable to compound errors to a large extent. Additionally, their sequential nature introduces additional delays that make maneuvering at high speeds and with agility difficult. Although these issues can be mitigated to some extent by manual tuning with expert knowledge, the divide-and-conquer principle that prevails in autonomous flight research in unknown environments commonly imposes fundamental limits on the speed and agility that flying robotic systems can achieve.

2.2. Imitation-Learning-Based Algorithms

Imitation-learning-based agents learn how to navigate by observing the trajectories of human experts or other robots that have completed specific tasks. Typically, a large volume of observational data is collected and used to train a neural network policy that can replicate an expert's decision-making process. The policy then predicts the next action to be taken from the input observation data and achieves the navigation goal by executing those actions. Imitation-learning-based algorithms are simple to train and, with sufficient training data, robots can learn how to navigate on their own. However, if the training data are insufficient or noisy, the policy may fail to make an optimal decision. Additionally, since the algorithm learns and selects actions based on existing data, it may be unable to handle situations it has never seen before. Typical published studies [18,27,28] used an imitation learning algorithm to train a policy as closely as possible to the expert's behavior. However, the policy was heavily dependent on the input experience.

2.3. Deep-Reinforcement-Learning-Based Algorithms

Recently, research on end-to-end robot navigation using DRL has become increasingly popular. Yarats et al. [24] proposed a SAC_AE policy with regularization constraints on the decoder loss. Then, in [29], Huang et al. used the regularized SAC_AE policy (SAC_RAE) to complete a distributed multiUAV collision avoidance task, where the flying robots were able to avoid each other and reach the target point using only the depth image from a front-facing deep camera. However, the validity of this policy was not well-demonstrated, because the experiment was conducted in an unobstructed open space. Following the success of the transformer [30] in the CV field, a combination of the transformer and reinforcement learning has been proposed in several works [31–33]. In these works, transformers were used to extract feature information from observation, which was then input into the policy network for learning, thereby achieving satisfactory results in their task scenarios.

However, we have noticed that the introduction of transformer modules in DRL may make policy training more challenging. Nevertheless, the literature suggests that it is theoretically possible to use vision transformers to build an encoder network for the perception module, which takes in all observation information (including depth images and agent state information), extracts latent variables, and computes the attention between them. In practice, transformer modules may lead to unstable learning, particularly in situations in which the agent's action set is rich and continuous. Therefore, to address this issue, we explored methods to increase the receptive field of convolutional modules, rather than relying solely on the large receptive field advantage of transformer modules.

3. Methodology

3.1. Problem Formulation

The objective of this study was to enable a flying robot to navigate rapidly in complex and unpredictable wild environments using DRL. The agent was limited to receiving only the depth image within a 15-m range ahead (consistent with the ZED Mini stereo camera) and its own pose information, thereby leading to restricted observation of the interaction process with the environment. Therefore, this study falls under the category of POMDP (Partially Observable Markov Decision Process), described as a 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, \gamma)$. Therein, the state space $\mathcal{S}$ represents the hidden variable, $\mathcal{A}$ is the action space, function $\mathcal{T}(s^{'}|s,a)$ denotes the state transition probability, $\mathcal{R}$ is the reward function, $\mathcal{O}$ is the observation space, γ is the discount factor, and $\Omega(o|s,a)$ represents the observation probability.

At the beginning of each episode, target points are randomly generated in an interactive environment with a constant Euclidean distance. The episode ends only when the target point is reached, timed out, or when the distance to the obstacle is less than that of the safe value. Specifically, at each time step t, the flying robot obtains a frame for visual observation o_z^t and a frame for pose observation o_p^t. It then executes action a^t, and it obtains the environment's reward r^t. By repeating this process, the strategy guides the agent to avoid obstacles and reach the target point by outputting the desired action.

3.2. Policy Setup

3.2.1. Observation Space

The pose observation o_p consists of three parts: $o_p = [o_g, o_v, o_a]$. Therein, o_g denotes the target position information in the body coordinate system, o_v denotes its own velocity observation, and o_a represents the acceleration observation. The visual observation o_z includes four consecutive 160 × 120 depth images at adjacent moments. At each time step t, the observation information obtained by the agent is $o^t = [o_z^t, o_p^t]$.

3.2.2. Action Space

Our agent enjoys complete control freedom without explicit limitations on its action space. At each time step t, our policy outputs an action command consisting of four degrees of control: v_x, v_y, v_z, and yaw_rate. v_x, represents velocity in the x direction of the body coordinate system for the scaling factor $k \in (0, 4)$, and it has a value range of $k \cdot 3 \cdot [-0.2, 1.0]$ m/s. Similarly, v_y and v_z representing velocity in the y and z directions, respectively, and they have a value range of $k \cdot 2 \cdot [-1.0, 1.0]$ m/s, while yaw_rate represents the rate of change in the heading angle of the agent, and it has a value range of $k \cdot 1.6 \cdot [-1.0, 1.0]$ rad/s; k is always equal to 1 in our experiments.

3.2.3. Reward Setup

Sparse rewards can make it challenging for reinforcement learning algorithms to learn, because the agent must undergo multiple trial-and-error iterations to discover the right course of action. However, incorporating auxiliary rewards can facilitate learning the correct policy, thus leading to a faster and more efficient learning process. Our reward function is composed of two distinct components. First, the r_g guides the agent to complete the primary task. The second component, r_a, assists the agent in learning a policy to expedite the achievement of this objective and penalizes the agent for outputting an inefficient action to encourage the generation of a more efficient output. At time step t, the reward function can be described as follows:

$$r^t = r_g^t + r_a^t \tag{1}$$

For a basic reward, R_b ($R_b \in \mathbb{N}_+$), r_g^t, and r_a^t are calculated as follows:

$$r_g^t = \begin{cases} R_b & \text{if } \|g^t\| < 1.0 \\ -R_b & \text{if collision occurs} \end{cases} \tag{2}$$

$$r_a^t = R_b \cdot (\frac{\|g^{t-1} - g^t\|}{g^0} - \frac{\alpha}{t^{max}} - \frac{\beta}{t^{max}} \cdot M) \tag{3}$$

The α parameter encourages the flying robot to reach its target point as quickly as possible, either by taking a shorter path or by moving at a faster speed. The β parameter motivates the flying robot to reduce the angle between the velocity vector direction and the visual observation direction to reduce unnecessary blind flight adventures of the flying robot. For example, a flying robot may choose to climb directly vertically to the top of an obstacle and then move toward a target point to avoid the obstacle, or the flying robot may appear to fly sideways to the left or right or backwards so that it is not able to visually observe movement in the direction. The settings for the α and β parameters should be between 0 and 1. The M parameter is used to determine whether the flying robot has abnormal movements, such as unnecessary blind flight adventures; if the strategy outputs an abnormal action, then M = 1. g^0 represents the Euclidean distance between the starting point and the target point, $\|g^{t-1} - g^t\|$ represents the Euclidean distance of the straight-line distance that the agent moves to the target point at the current moment, and t^{max} represents the maximum time steps of actions that can be executed in each round of episode.

In our experiments, we set α, β, and R_b to 0.1, 0.5, and 20, respectively.

3.3. Network Architecture

Our method flow chart is shown in Figure 2. Our method consists of three key modules, i.e., the vision encoder, the state encoder, and the reinforcement learning (RL) policy module. Two encoders extract critical information and project it onto a feature vector on depth images and IMU information, respectively. After that, two extracted features are concatenated and fed into a reinforcement learning policy module for strategy learning. According to the current environment, the RL module returns a set of actions for UAV control, including velocity on three dimensions and yaw rate.

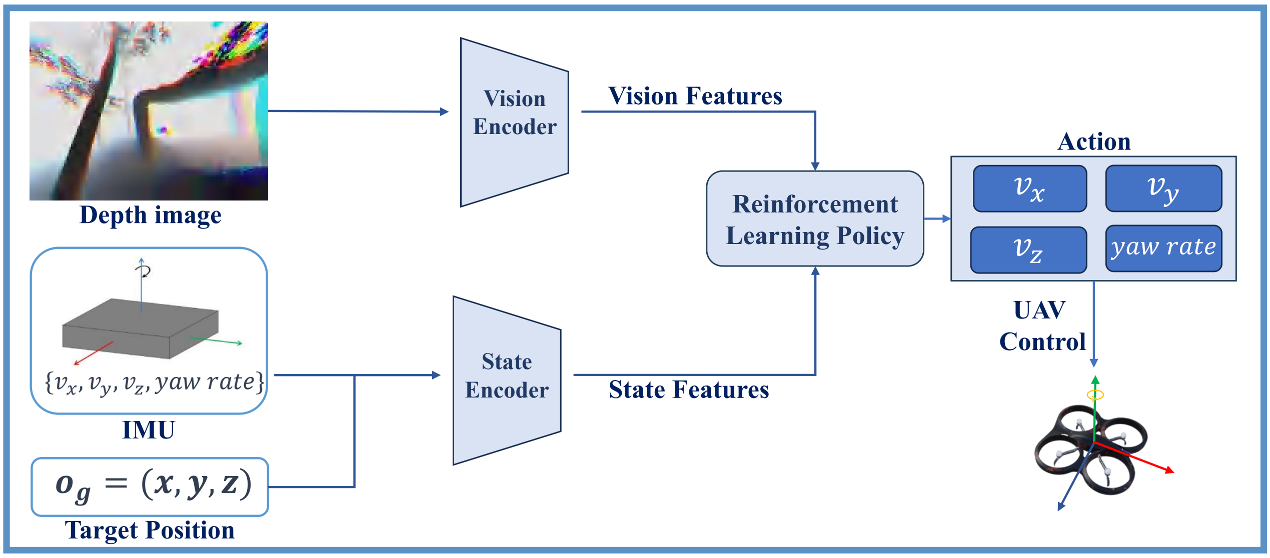

Figure 2. Method overview.

Our SAC_FAE network structure, as shown in Figure 3, employs convolution to acquire latent variables from visual observations. However, we did not simply utilize plain convolutional autoencoders. Instead, we utilized a focus module to execute a unique type of convolution operation. The focus module was originally embodied in YOLO v5 and is featured in the use of the number of channels in exchange for the resolution of the feature map. This operation groups the input tensors by channel, arranges pixels in each channel in a particular order, and performs ordinary convolution operations. This grouping and rearrangement process effectively decreased the amount of computation and memory consumption required by the focus module. Specifically, we used a slicing operation to divide a high-resolution feature map into multiple low-resolution feature maps and concatenated them on channels.

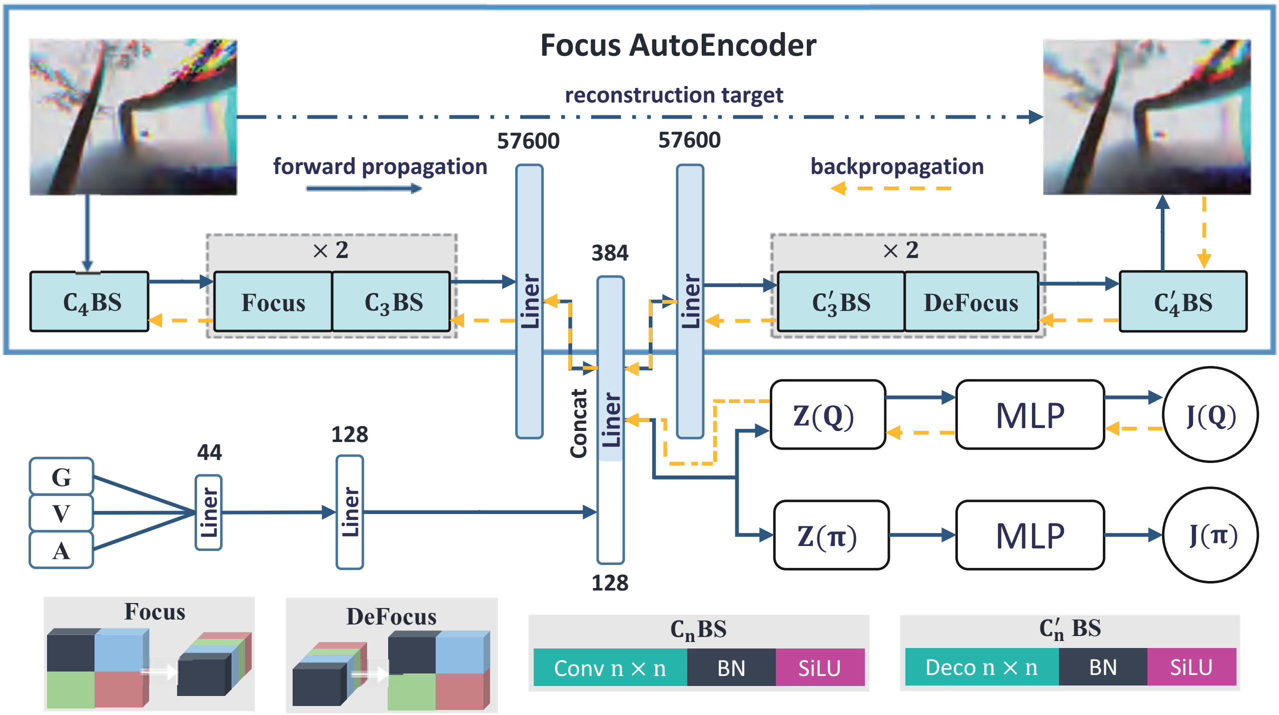

Figure 3. Schematic diagram of SAC_FAE network structure. Each linear layer is followed by a SiLU activation function, where Conv represents the convolutional layer, Deco represents the deconvolution layer, and BN stands for batch normalization. The measurement feature and encoder feature are concatenated in the last dimension and transferred to the multilayer perception (MLP) layer of the actor and critic networks.

This approach enabled us to complete downsampling operations without losing information and enhance the feature expression capacity of the proposed model through a larger receptive field. Although we can obtain lossless feature information and enhance the global receptive field through the focus operation, much of the feature information is still redundant and does not help in model training. Therefore, after each focus operation, we used a 3 × 3 convolutional layer to reduce the number of channels to half of the original. Simultaneously, we truncated the gradients from the actors to avoid the impact of the exploration process on the uncertainty of the encoder learning.

The encoder extracted 384 latent variable features (encoder feature dim) from the input image observations. For the pose and target point of the flying robot, 128 latent variable features (measurement feature dim) were obtained after two layers of MLP. Both the actor and critic had a 4-layer multilayer perception (MLP) with 1024 units.

3.4. Training Strategies

To improve the sample efficiency, we utilized experience replay buffers to store the environment interaction trajectories. We discovered that the hyperparameters of the buffer capacity and training frequency play an important role in determining the final outcome. Setting the buffer capacity too high can impede the learning process and increase model complexity, because there will be differences between the data generated before and after the policy interacts with the environment. When the replay buffer is set too small, it leads to unstable training, and the results tend to fall into local optima. Therefore, the buffer capacity was set to store approximately 50 interaction fragments and train them at the end of each round. The pseudocode for our policy-training process is presented in Algorithm 1.

Algorithm 1: Policy training

```
Initialize environment, policy, and replay buffer $\mathcal{B}$;
for $i$ = 0 to max episodes do
    for $j$ = 0 to max timesteps do
        if not terminal then
            Select action $a^j$;
            Interact with environment;
            Receive $r^j$, $o^{j+1}$, terminal;
            Store [$o^j$, $a^j$, $r^j$, $o^{j+1}$, terminal] to $\mathcal{B}$;
        end
    end
    while cnt < 200 do
        Randomly sample batches of trajectories [$o$, $a$, $r$, $o^{next}$, terminal] from $\mathcal{B}$;
        update policy network;
    end
end
```

4. Experimental Setup

4.1. Simulation Assessment

We built multiple simulation environments in Unreal Engine 4, and we used the AirSim plugin to complete the flight simulation of the flying robot. The flying robot was trained on a computer equipped with an Intel i9-11900k, NVIDIA RTX A4000, with 64 GB of RAM. A policy network was constructed using the Pytorch library. The frequency for obtaining the depth image of the flying robot was 10 Hz (160 × 120), the control frequency was 20 Hz (main loop frequency), and the camera field of view was 120°. In all experiments, the effective range of the depth image was [0.1, 15.0] meters, which is consistent with the popular ZED Mini stereo camera. Gaussian noise with a mean of 0 and a variance of 0.1 was added to the acquired true depth image. Using home as the origin, before the start of each episode, a point on a circle with a radius of 20.0 m was randomly selected as the target point.

4.2. Policy Hyperparameter

The hyperparameters we set during policy training are listed in Table 1. The listed hyperparameters can be divided into three categories. First, hyperparameters of the optimizer for model training were used. Second, hyperparameters of the feature extraction procedure were used. Here, we considered the feature dimensions of the visual feature and state feature, respectively. Third, hyperparameters of the reinforcement learning procedure were used.

Table 1. Hyperparameters.

Item Name	Value
Activation	SiLU
Optimizer	Adam
Learning rate	3×10^{-4}
Encoder feature dim	384
Measurement feature dim	128

Table 1. *Cont.*

Item Name	Value
Max time steps	400
Replay buffer B capacity	25,600
Batch size	256
Gamma	0.95
Actor update frequency	1
Actor log stddev bounds	$[-10, 2]$
Critic target update frequency	2
Critic encoder soft-update rate	0.025
Critic Q-function soft-update rate	0.005

4.3. Performance Metrics

To reflect the effectiveness and robustness of the strategy over N consecutive episodes (N = 50), we used the following performance indicators:

- Success rate (SR): The percentage of flying robots that successfully reached the corresponding target point without collision in limited time step.
- Crash rate (CR): The percentage of total experiments in which the distance between the flying robot and the obstacle was less than the safety value during the navigation process.
- Success rate weighted by path length (SPL): The proportion of robots that successfully reached the target location, considering the path length, which is calculated as follows:

$$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(p_i, l_i)} \tag{4}$$

  where S_i is a binary symbol marking the success of the ith episode, p_i refers to the actual flight trajectory length of the robot in the ith episode, and l_i is the Euclidean distance from the home point to the target point in the ith episode.
- Average Speed: The average speed of the flying robot, which is obtained from the average speed of each episode.
- Max Speed: The maximum speed of the flying robot.

4.4. Experimental Steps

To verify the generalizability of the policy, we conducted training within a single scenario and evaluated its performance across various scenarios. During experiments, we implemented all methods in our simulation systems following their original implementations. The simulation settings were kept consistent during all experiments, including resolution and effective distance of depth images, frequency of controller reaction, and image acquisition. For the setup of the experimental scene, we simulated three testing scenarios with different difficulties by adjusting the obstacle densities (0.1 m^{-3}, 0.5 m^{-3}, and 0.9 m^{-3} for simple, medium, and complex scenarios, respectively), which can be clearly observed in Figure 4. After training in Scene 2, our strategy underwent multiple rounds of performance metric calculations in all three scenes to exhibit its robustness and effectiveness.

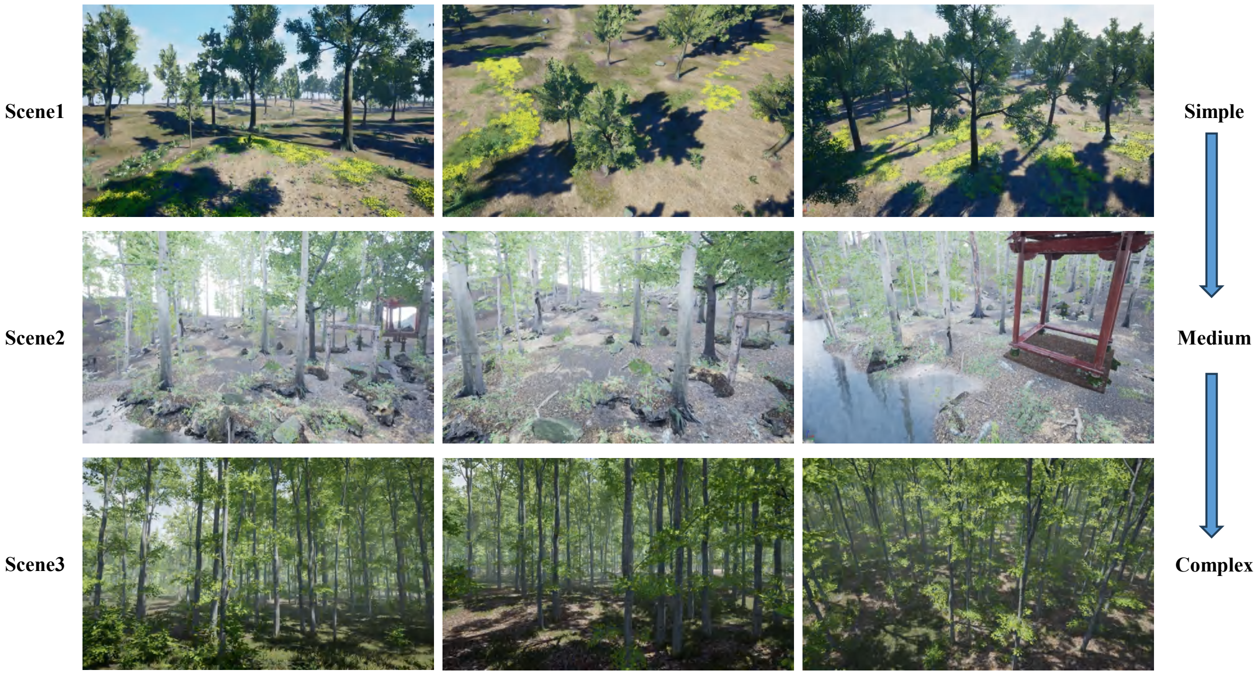

Figure 4. Experimental scenes.

5. Results

Here, the SAC_RAE [29] was selected for comparison, since it is the first work to achieve distributed multiUAV collision avoidance using deep reinforcement learning (DRL) techniques. In addition, since we focused on improving the visual feature extraction in this work, we also replaced the vision model of the SAC_RAE with two popular and strong architectures, i.e., CNN and ViT, to construct two stronger variants for comparison, i.e., SAC_CNN and SAC_ViT. In addition, we also selected two very popular traditional trajectory generation optimization methods—Fast-Planner [25] and EGO-Planner [26]—for comparison. We trained all of the DRL-based policies for 1200 episodes, and we measured their performance outcomes using the aforementioned metrics in three different scenarios, as shown in Table 2 and Figure 5. From Table 2, we have three observations. First, from simple to complex scenarios, the performance of all methods became reduced, which is reasonable, since it is difficult for autonomous navigation to take place in a complex environment. Second, our method could outperform previous state-of-the-art (SOTA) methods in most settings, with both a higher success rate and speed. Third, the proposed model performed well in the timeout metric across multiple scenarios, which reflects the decisiveness of the strategy in situations where multiple optimal solutions may exist.

Table 2. Experimental results.

Method	Scene 1 (Simple)					Scene 2 (Medium)					Scene 3 (Complex)				
	Flight Quality			Speed (m/s)		Flight Quality			Speed (m/s)		Flight Quality			Speed (m/s)	
	SR	CR	SPL	Avg	Max	SR	CR	SPL	Avg	Max	SR	CR	SPL	Avg	Max
FAST-Planner [25]	0.10	0.32	0.10	1.64	3.07	0.16	0.34	0.14	1.80	3.43	0.10	0.74	0.10	**2.40**	3.36
EGO-Planner [26]	0.80	0.18	**0.76**	1.84	2.84	0.65	**0.25**	0.60	2.20	2.87	**0.42**	0.50	**0.36**	1.60	3.36
SAC_RAE [29]	0.80	0.16	0.66	1.96	3.32	0.62	0.38	0.60	2.36	3.38	0.14	0.86	0.13	2.29	3.15
SAC_CNN	0.56	0.44	0.44	2.15	3.18	0.48	0.52	0.43	**2.39**	3.32	0.20	0.80	0.18	2.37	3.19
SAC_ViT	0.00	0.26	0.00	0.86	1.12	0.00	0.02	0.00	0.85	1.11	0.00	**0.20**	0.00	0.86	1.12
SAC_FAE (Ours)	**0.82**	**0.12**	0.71	**2.37**	**3.54**	**0.74**	0.26	**0.71**	2.34	**3.52**	0.26	0.74	0.23	2.27	**3.47**

The numbers in bold represent the optimal data for this column.

In the absence of pretrained weights, the SAC_ViT agent, based on its transformer architecture, interacted with the environment to learn. Although this model possesses a low crash rate compared to the other models, this is due to the fact that the model has only mastered basic machine control, and its success rate was the lowest across scenarios. Our analysis suggests that self-attention mechanisms require extensive learning and even prior knowledge, which are similar to the approach described in [32], where they first employed imitation learning from a large number of expert trajectories to endow their strategy with some exploratory ability from the outset. However, long-term training or the provision of expert trajectories is expensive and time-consuming. Under the same experimental settings as those of the proposed method, the SAC_ViT performance remained significantly improved.

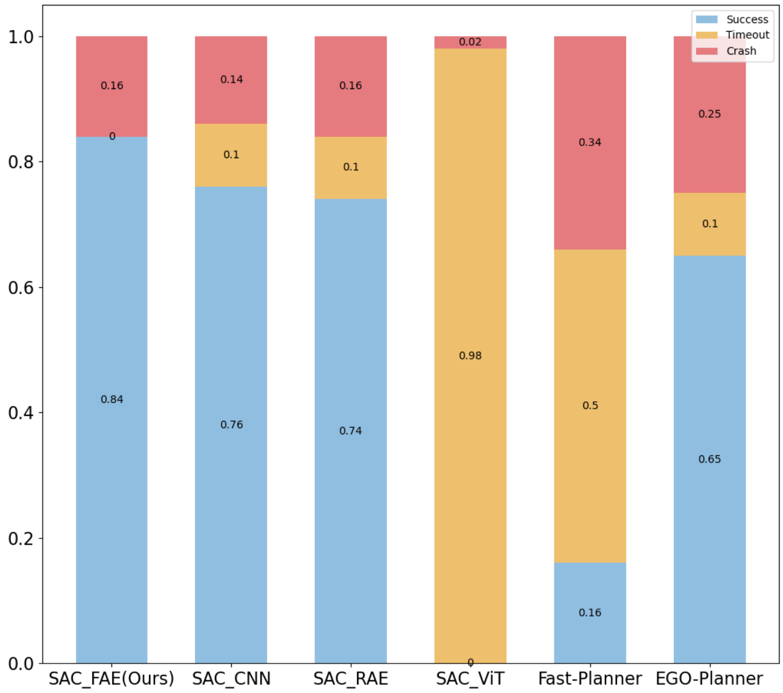

Figure 5. The comparison result in Scene 1.

In the SAC_RAE architecture, this strategy performs multiple downsampling procedures on the projection feature map in the autoencoder, which is prone to a loss of obstacle position information, and the probability of a collision or timeout will increase. As is shown in Figure 6, the average reward of this architecture fluctuated significantly during exploration.

Within the SAC_CNN architecture, only the gradient of the critic network was utilized in the perception module of the policy, thus resulting in the fastest initial learning speed. However, due to the instability of the critic during exploration, significant fluctuations can occur. As a result, the convergence performance of this policy was comparable to that of the SAC_RAE.

In addition, it is worth noting that the Fast-Planner algorithm was also effective, and its timeout cases were usually due to modifications of the target position made by the planner to ensure a better hovering attitude. Actually, in Fast-Planner, the inability to place control points around obstacles presents a significant hindrance in dense obstacle environments for flying robots; its selection of the control points avoids obstacles as much as possible, but camera noise can sometimes lead to false perceptions of planned trajectories or false perceptions as to whether the control points are within obstacles, thus leading to replanning. In practice, the flying robots deployed with the Fast-Planner algorithm typically came within 2.5 m of the expected target point in around 66% of the cases. The actual arrival at the target point itself occurred only rarely (though open scenes could facilitate this). As our experiment considered navigation success only when the robot's Euclidean distance to its target was under 1.0 m, this prevented Fast-Planner from presenting the same performance as EGO-Planner in the success rate metric.

EOG-Planner achieved a high success rates in most scenarios, especially in Scene 3, where the success rate of EGO was higher than all the methods based on reinforcement learning. The reason is that the simulation environments can only render depth images at 10 Hz, which is significantly insufficient for fast flight in a complex environment. If the camera signal is lost, the learned reinforcement learning strategy will sample a wrong action, thereby leading to a higher crash rate. However, EGO-Planner modeled the UAV dynamics and maintained the local map information. If the camera signal is lost, the traditional planner can obtain the environment information from the built local map and achieve the navigation.

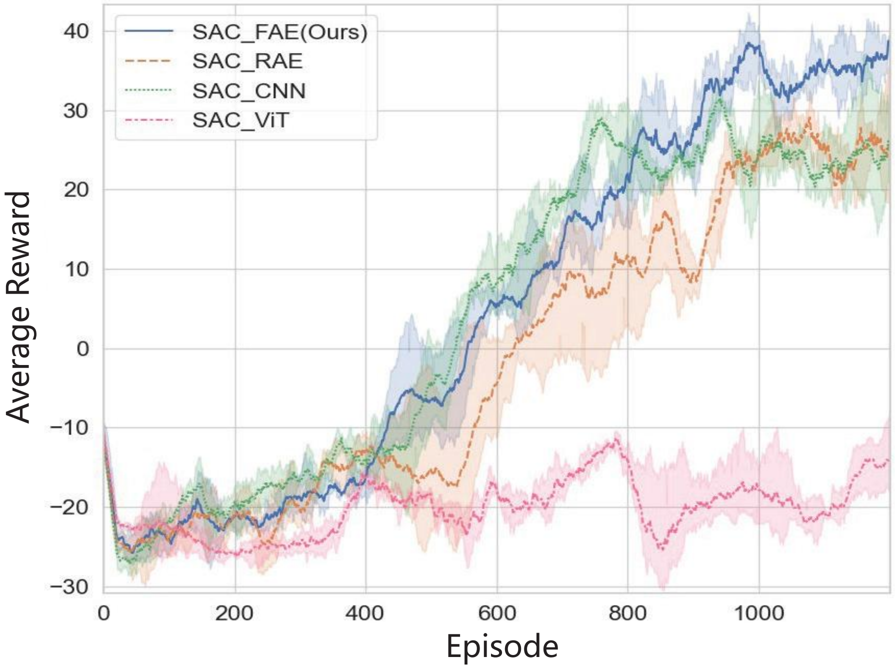

Figure 6. A visual comparison result of the average return while agents interacted with the experimental natural forest (blue line for Ours).

For the DRL-based strategies, we also display the average return during the training process in Figure 6, where the shaded area represents the variance; it accounts for the stability of the policy-training process.

6. Conclusions

We presented an effective focus autoencoder module in this study, which performed the lossless downsampling of feature maps through slicing operations and achieved a larger receptive field. Experimental results show that our method could outperform previous SOTA methods in most settings, with both a higher success rate and speed. To demonstrate the effectiveness of our strategy, we conducted multiple experiments ranging from simple to complex scenarios in different complex environments. Our proposed method outperformed in multiple test environments, thereby exhibiting good robustness.

Although we believe that there is great potential for combining deep reinforcement learning with transformers, training intelligent agents based on transformers is challenging. We need to continuously develop more effective methods to promote community growth and enable autonomous flying robots based on strong reinforcement learning to be sufficiently powerful.

Author Contributions: Conceptualization, Z.F.; methodology, H.L.; software, K.H. and H.L.; validation, K.H. and H.L.; formal analysis, K.H.; investigation, K.H. and H.L.; resources, K.H. and H.L.; data curation, K.H.; writing—original draft preparation, K.H. and H.L.; writing—review and editing, K.H., H.L., J.Z., Z.H. and Z.F.; visualization, K.H. and H.L.; supervision, Z.H.; project administration, K.H., H.L., J.Z. and Z.F.; funding acquisition, J.Z. and Z.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the National Key R&D Program of China (grant numbers 2021ZD0111501 and 2021ZD0111502), the Key Laboratory of Digital Signal and Image Processing of Guangdong Province, the Key Laboratory of Intelligent Manufacturing Technology (Shantou University), the Ministry of Education, the Science and Technology Planning Project of Guangdong Province of China (grant number 180917144960530), the Project of Educational Commission of Guangdong Province of China (grant number 2017KZDXM032), the State Key Lab of Digital Manufacturing Equipment and Technology (grant number DMETKF2019020), the National Natural Science Foundation of China (grant number 62176147), the STU Scientific Research Foundation for Talents (grant number NTF21001, NTF22030), the Science and Technology Planning Project of Guangdong Province of China (grant number 2019A050520001, 2021A0505030072, 2022A1515110660), the Science and Technology Special Funds Project of Guangdong Province of China (grant number STKJ2021176, STKJ2021019), the Guangdong Special Support Program for Outstanding Talents (2021JC06X549), and the Li Ka Shing Foundation Cross Research Project (2020LKSFG02D).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

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

Abbreviations

The following abbreviations are used in this manuscript:

DRL	Deep Reinforcement Learning
UAV	Unmanned Aerial Vehicle
SAC	Soft Actor–Critic
FAE	Focus Autoencoder
RAE	Regularized Autoencoder
ESDF	Euclidean Signed Distance Field
POMDP	Partially Observable Markov Decision Process
MLP	Multilayer Perception
CNN	Convolutional Neural Network
ViT	Vision Transformer

References

1. Christiansen, M.P.; Laursen, M.S.; Jørgensen, R.N.; Skovsen, S.; Gislum, R. Designing and testing a UAV mapping system for agricultural field surveying. *Sensors* **2017**, *17*, 2703. [CrossRef]
2. Leduc, M.B.; Knudby, A.J. Mapping wild leek through the forest canopy using a UAV. *Remote Sens.* **2018**, *10*, 70. [CrossRef]
3. Kim, S.; Kim, D.; Jeong, S.; Ham, J.W.; Lee, J.K.; Oh, K.Y. Fault diagnosis of power transmission lines using a UAV-mounted smart inspection system. *IEEE Access* **2020**, *8*, 149999–150009. [CrossRef]
4. Tian, Y.; Liu, K.; Ok, K.; Tran, L.; Allen, D.; Roy, N.; How, J.P. Search and rescue under the forest canopy using multiple UAVs. *Int. J. Robot. Res.* **2020**, *39*, 1201–1221. [CrossRef]
5. Chen, Z.; Alonso-Mora, J.; Bai, X.; Harabor, D.D.; Stuckey, P.J. Integrated task assignment and path planning for capacitated multi-agent pickup and delivery. *IEEE Robot. Autom. Lett.* **2021**, *6*, 5816–5823. [CrossRef]
6. Loquercio, A.; Kaufmann, E.; Ranftl, R.; Dosovitskiy, A.; Koltun, V.; Scaramuzza, D. Deep drone racing: From simulation to reality with domain randomization. *IEEE Trans. Robot.* **2019**, *36*, 1–14. [CrossRef]
7. Ju, C.; Son, H.I. Modeling and control of heterogeneous agricultural field robots based on Ramadge–Wonham theory. *IEEE Robot. Autom. Lett.* **2019**, *5*, 48–55. [CrossRef]
8. Wu, K.; Wang, H.; Esfahani, M.A.; Yuan, S. Learn to navigate autonomously through deep reinforcement learning. *IEEE Trans. Ind. Electron.* **2021**, *69*, 5342–5352. [CrossRef]

9. Hu, H.; Zhang, K.; Tan, A.H.; Ruan, M.; Agia, C.; Nejat, G. A sim-to-real pipeline for deep reinforcement learning for autonomous robot navigation in cluttered rough terrain. *IEEE Robot. Autom. Lett.* **2021**, *6*, 6569–6576. [CrossRef]
10. Oleynikova, H.; Burri, M.; Taylor, Z.; Nieto, J.; Siegwart, R.; Galceran, E. Continuous-time trajectory optimization for online uav replanning. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Republic of Korea, 9–14 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 5332–5339.
11. Ding, W.; Gao, W.; Wang, K.; Shen, S. An efficient b-spline-based kinodynamic replanning framework for quadrotors. *IEEE Trans. Robot.* **2019**, *35*, 1287–1306. [CrossRef]
12. Zhou, B.; Pan, J.; Gao, F.; Shen, S. Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Trans. Robot.* **2021**, *37*, 1992–2009. [CrossRef]
13. Oleynikova, H.; Taylor, Z.; Fehr, M.; Siegwart, R.; Nieto, J. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1366–1373.
14. Han, L.; Gao, F.; Zhou, B.; Shen, S. Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 4423–4430.
15. Ratliff, N.; Zucker, M.; Bagnell, J.A.; Srinivasa, S. CHOMP: Gradient optimization techniques for efficient motion planning. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 489–494.
16. Gao, F.; Lin, Y.; Shen, S. Gradient-based online safe trajectory generation for quadrotor flight in complex environments. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 3681–3688.
17. Ishihara, K.; Kanervisto, A.; Miura, J.; Hautamaki, V. Multi-task learning with attention for end-to-end autonomous driving. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; pp. 2902–2911.
18. Loquercio, A.; Kaufmann, E.; Ranftl, R.; Müller, M.; Koltun, V.; Scaramuzza, D. Learning high-speed flight in the wild. *Sci. Robot.* **2021**, *6*, eabg5810. [CrossRef] [PubMed]
19. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.
20. Fan, T.; Long, P.; Liu, W.; Pan, J. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *Int. J. Robot. Res.* **2020**, *39*, 856–892. [CrossRef]
21. Xue, Y.; Chen, W. A UAV Navigation Approach Based on Deep Reinforcement Learning in Large Cluttered 3D Environments. *IEEE Trans. Veh. Technol.* **2022**, *72*, 3001–3014. [CrossRef]
22. Zhang, S.; Li, Y.; Dong, Q. Autonomous navigation of UAV in multi-obstacle environments based on a Deep Reinforcement Learning approach. *Appl. Soft Comput.* **2022**, *115*, 108194. [CrossRef]
23. Barth-Maron, G.; Hoffman, M.W.; Budden, D.; Dabney, W.; Horgan, D.; Dhruva, T.; Muldal, A.; Heess, N.; Lillicrap, T. Distributed Distributional Deterministic Policy Gradients. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April 30–3 May 2018.
24. Yarats, D.; Zhang, A.; Kostrikov, I.; Amos, B.; Pineau, J.; Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; pp. 10674–10681.
25. Zhou, B.; Gao, F.; Wang, L.; Liu, C.; Shen, S. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robot. Autom. Lett.* **2019**, *4*, 3529–3536. [CrossRef]
26. Zhou, X.; Wang, Z.; Ye, H.; Xu, C.; Gao, F. Ego-planner: An esdf-free gradient-based local planner for quadrotors. *IEEE Robot. Autom. Lett.* **2020**, *6*, 478–485. [CrossRef]
27. Karnan, H.; Warnell, G.; Xiao, X.; Stone, P. Voila: Visual-observation-only imitation learning for autonomous navigation. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 2497–2503.
28. Watkins-Valls, D.; Xu, J.; Waytowich, N.; Allen, P. Learning your way without map or compass: Panoramic target driven visual navigation. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2021; IEEE: Piscataway, NJ, USA, 2020; pp. 5816–5823.
29. Huang, H.; Zhu, G.; Fan, Z.; Zhai, H.; Cai, Y.; Shi, Z.; Dong, Z.; Hao, Z. Vision-based Distributed Multi-UAV Collision Avoidance via Deep Reinforcement Learning for Navigation. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 13745–13752.
30. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *arXiv* **2017**, arXiv:1706.03762.
31. Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 15084–15097.

32. Huang, W.; Zhou, Y.; He, X.; Lv, C. Goal-guided Transformer-enabled Reinforcement Learning for Efficient Autonomous Navigation. *arXiv* **2023**, arXiv:2301.00362.
33. Esslinger, K.; Platt, R.; Amato, C. Deep Transformer Q-Networks for Partially Observable Reinforcement Learning. *arXiv* **2022**, arXiv:2206.01078.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.