Article

Learning to Have a Civil Aircraft Take Off under Crosswind Conditions by Reinforcement Learning with Multimodal Data and Preprocessing Data

Feng Liu ^{1,*} , Shuling Dai ^{1,2} and Yongjia Zhao ^{1,2}¹ State Key Laboratory of VR Technology & Systems, Beihang University, Beijing 100191, China; sldai@buaa.edu.cn (S.D.); zhaoyongjia@buaa.edu.cn (Y.Z.)² Jiangxi Research Institute, Beihang University, Beijing 100191, China

* Correspondence: by1503101@buaa.edu.cn

Abstract: Autopilot technology in the field of aviation has developed over many years. However, it is difficult for an autopilot system to autonomously operate a civil aircraft under bad weather conditions. In this paper, we present a reinforcement learning (RL) algorithm using multimodal data and preprocessing data to have a civil aircraft take off autonomously under crosswind conditions. The multimodal data include the common flight status and visual information. The preprocessing is a new design that maps some flight data by nonlinear functions based on the general flight dynamics before these data are fed into the RL model. Extensive experiments under different crosswind conditions with a professional flight simulator demonstrate that the proposed method can effectively control a civil aircraft to take off under various crosswind conditions and achieve better performance than trials without visual information or preprocessing data.

Keywords: autopilot; civil aircraft; multimodal data; reinforcement learning; preprocessing

Citation: Liu, F.; Dai, S.; Zhao, Y. Learning to Have a Civil Aircraft Take Off under Crosswind Conditions by Reinforcement Learning with Multimodal Data and Preprocessing Data. *Sensors* **2021**, *21*, 1386. <https://doi.org/10.3390/s21041386>

Academic Editor: Wai Lok Woo

Received: 14 December 2020

Accepted: 10 February 2021

Published: 16 February 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The autopilot function of civil aircraft is a technology that allows the aircraft to control itself and complete some flying tasks autonomically, and this technology has been developed over decades. The traditional autopilot function of civil aircraft uses positional data and attitude data usually acquired from sensors in the aircraft to control the plane and guarantee it flies as planned. Examples of tasks accomplished via autopilot include tracking an airline, maintaining smooth flight, maintaining a given airspeed or altitude, and automatically landing by the guidance of an instrument landing system [1–4].

Although the current autopilot system of civil aircraft has been able to accomplish many flying tasks, it becomes ineffective under severe turbulence or dangerous weather, and at some critical phases, such as taking off and landing, pilots are unwilling to use the autopilot function [5–7]. The reason why the traditional autopilot function is not adequate under some emergency or critical conditions is because the traditional autopilot algorithm was designed manually based on aerodynamics, mechanics and control science. However, the abnormal flying conditions are too complex to model manually [8]. As a result, the traditional autopilot function of civil aircraft works only on some simple tasks under calm and common conditions.

The automatic driving of a car based on machine learning has been studied for many years. Some studies use deep learning to process visual data and radar data to improve the cognitive ability of autonomous driving programs on road conditions [9,10]. In order to get a program that can navigate and control a car without using expert data, some studies use reinforcement learning to learn automatically in the environment [11,12]. Civil aviation, as a mode of flying transportation, is very similar to car traffic. Therefore, it is practical and significant to use the most advanced machine learning technology to improve the autopilot capacity of civil aircraft.

A few studies dedicated to the realization of the auto flying of aerial vehicles through advanced machine learning algorithms have been performed in recent years. NASA implemented a project named L2F that used a modified MiG-27 foam target drone and some sensors to conduct real-time aerodynamic modeling and to learn adaptive control [13]. In 2018, Anwar and Raychowdhury successfully made an unmanned aerial vehicle (UAV) learn to fly in a real environment via end-to-end deep reinforcement learning using monocular images [14]. Shaker and Smith presented a fast reinforcement learning algorithm for an unmanned aerial vehicle to learn how to automatically land using visual information [15]. However, most of these works focused on diminutive UAVs and quadcopters, which have simpler structures and are easier to control than complicated and sluggish civil aircraft. In this paper, we focus on a tough task — learning to have a civil aircraft take off under crosswind conditions. Another difference from the previous works is that we use more information to learn to fly. We provide multimodal state data for learning, including real-time flight status data and visual data while the previous works use single-model data; at the same time, we provide preprocessing data that are designed based on general flight dynamics.

Because supervised learning requires a large number of demonstrated samples [16,17] and it is difficult to employ a professional pilot to fly and collect an adequate number of samples, in this study, we use reinforcement learning (RL) [18], which is an unsupervised learning method, to identify the aerodynamics of civil aircraft without a prior aerodynamic knowledge; then, using the RL model to stabilize, control, and navigate the aircraft to accomplish the take-off process. Finally, we perform experiments with a professional flight simulation environment, and the results demonstrate that our machine learning model is able to master this challenging task.

The remainder of the paper is organized as follows—in Section 2, we introduce a few related works. Section 3 introduces the technical background of this paper. In Section 4, we detail the proposed methodology. In Section 5, we describe detailed experimental settings, the results and discussions. In Section 6, we draw a conclusion from our research, discuss a drawback of the proposed method and describe our future studies.

2. Related Works

In 2018, Anwar navigated an unmanned aerial vehicle in an indoor real environment via end-to-end reinforcement learning. They used monocular images obtained from the camera in the aircraft as the state information for the RL model. They used double DQN [19] which is a classical RL algorithm to conduct this study and they used the depth of the image to generate the reward. To address safety issues, they created a virtual collision environment to train the aircraft first, and then completed the training in the real environment [14].

In 2019, Koch and Mancuso used reinforcement learning to control the attitude of a quadrotor in an open source high-fidelity simulation environment and utilized digital twinning concepts for minimal effort when transferring trained controllers to hardware [20]. They performed experiments for state-of-the-art RL algorithms on aircraft attitude control, such as deep deterministic policy gradient (DDPG) [21], trust region policy optimization (TRPO) [22] and proximal policy optimization (PPO) [23]. The results show that controllers trained using PPO outperform PID control and are capable of exceptional performance.

In 2020, Xie and Peng presented a reinforcement learning method to make a UAV autonomously track and land on a moving platform. They proposed a partially observable Markov decision process (POMDP). In the POMDP model, the UAV automatically learns the landing maneuver by an end-to-end neural network, which combines the deep deterministic policy gradients algorithm and heuristic rules. They used the position and velocity data of the UAV and the platform as the state information of the RL model, and they used the distance between UAV and moving platform to calculate the immediate reward. The experiments were performed on the Modular Open Robots Simulation Engine. Compared

with the PID methods, their method shows good performance when the platform moves in a complex trajectory [24].

Most prior works focused on the small unmanned aerial vehicles and provided single mode data to the RL model. In this study, we try to have a civil aircraft take off autonomously under crosswind conditions by reinforcement learning. Due to the large size and complex mechanical structure of a civil aircraft, we use multi-modal data and preprocessed data to train the learning model. The goal of this work is to improve the ability of the autopilot of civil aircraft under abnormal weather conditions by providing more information and using a more complex RL architecture.

3. Technical Background

3.1. Reinforcement Learning and DDPG

Reinforcement learning is a branch of machine learning, which learns how to establish policies by exploring an environment without any instructions. RL follows the discounted Markov decision process (S, A, γ, P, r) [25]. Each action $a_t \in A$ will lead to a subsequent state s_{t+1} according to the transition probability $P(s_{t+1}|s_t, a_t)$ and acquire a reward $r(s_t, a_t)$. Using action, state and reward information, the model updates its parameters at each learning step.

The early RL algorithms use a Q-table to record values denoting the learned knowledge and are updated obeying the Behrman equation [26]. Q-learning [27] is a classic RL algorithm that uses a Q-table to record the learned policy. However, since the capacity of the Q-table is limited, if the dimensions of the state or action are excessively high or if a continuous task is being learned, there will be a large amount of transitional value data, and it is difficult for the Q-table to accommodate such a large quantity of data. In recent years, with the development of the artificial neural network (ANN), deep reinforcement learning (DRL), which is a combination of RL and ANN, was proposed [28,29]. DRL possesses both the perception ability of deep learning and the policy-making ability of reinforcement learning, so DRL is competent at performing large-dimension tasks and continuous tasks.

The algorithm proposed in this paper is based on the DDPG method, a popular DRL algorithm that uses an actor-critic structure and outputs deterministic actions rather than a policy distribution. DDPG exhibits outstanding performance on continuous learning tasks and uses 4 networks: an evaluation actor $\mu(s|\theta^\mu)$, an evaluation critic $Q(s, a|\theta^Q)$, a target actor $\mu'(s|\theta^{\mu'})$ and a target critic $Q'(s, a|\theta^{Q'})$. The evaluation networks are used to explore and learn, and the target networks are mainly used to give criticism to the evaluation networks. The weights of the target networks are updated by slowly tracking the evaluation networks: $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$ with $\tau \ll 1$. The evaluation critic $Q(s, a|\theta^Q)$ is optimized by minimizing the loss:

$$L(\theta^Q) = \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta, r_t \sim E} [(Q(s_t, a_t|\theta^Q) - y_t)^2], \quad (1)$$

where

$$y_t = r(s_t, a_t) + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{Q'})). \quad (2)$$

The evaluation actor $\mu(s|\theta^\mu)$ is updated by the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}. \quad (3)$$

3.2. Simulation Environment

In this study, we use X-Plane, a professional flight simulator, to conduct the experiments. X-Plane is equipped with functions of advanced flight dynamics simulation, instrument simulation, flight environment simulation and flight operation simulation. X-Plane has been certified by the Federal Aviation Administration of the United States and has been used by the world's leading defense contractors, air force and aircraft manufacturers for a variety of applications from flight training to conceptual design and flight testing, and

also can be used for recording flight experience, private flight license training, instrument flight training. X-Plane is convenient for communicating with external applications by sending flight status data and receiving control commands through the User Datagram Protocol (UDP) or a secondary development plugin.

X-Plane has been used in many studies. A research team of Central Connecticut State University's School of Engineering have used X-Plane to design a full-scale helicopter simulator [30]. Jirgl and Boril used X-Plane to obtain mathematical identification results of an aircraft model and analyse parameters of mathematical models of human behavior while flying an aircraft [31]. In 2014, Kaviyarasu and Senthil Kumar Simulated a flapping-wing unmanned aerial vehicle using X-plane [32]. Due to the high cost and slow iteration of aircraft design and testing in real environments, X-Plane has been used by many organizations in industry, such as Boeing, NASA, Cessna, Piper, Japan Airlines, and the American Federal Aviation Administration [33].

To communicate with X-Plane, we use X-Plane Connect [34], a plugin developed by NASA, to communicate between X-Plane and our program. The X-Plane Connect (XPC) Toolbox is an open source research tool used to interact with the X-Plane commercial flight simulator and allows users to control an aircraft and receive state information from X-Plane by communicating hundreds of flight data in real time. This research tool has been used to visualize flight paths, test control algorithms, and simulate an active airspace with various aircraft and airlines [35,36].

4. Methodology

4.1. State Information for Reinforcement Learning

The state information of the environment is referred to as a set of input data for the reinforcement learning model to make a determinative action based on its learned policy, and the components of the input data are critical to the effectiveness of the learning. In this study, the state information consists of 3 sections—(1) The common flight status data; (2) the preprocessing of some flight status data; and (3) visual data from the master pilot's perspective. Table 1 lists the detailed state information for this study.

4.1.1. Flight Status Data

The flight status of the position, velocity, control, and so forth, can be obtained from sensors in the aircraft or other equipment. Certainly, we cannot obtain these data from the sensors directly; rather, we obtain these data indirectly from the related embedded computer. We also normalize these data to make them more moderate for the neural network. In this study, the flight status we used are as follows:

- **Positional and rotational information:** The positional data include longitude, latitude and altitude, which are denoted as P_x , P_y and P_z , respectively, in this paper. Generally, this information can be obtained from GPS, ground-based augmentation system or air pressure sensors. The rotational data include the pitch, roll and heading of the aircraft, and these data are denoted as R_p , R_r and R_h , respectively.
- **Velocity information:** The velocity information of an aircraft includes V_x , V_y , V_z , V_p , V_r and V_h , which correspond successively to the 3 positional data variables and 3 rotational data variables.
- **True airspeed:** The true airspeed V_t , which represents the relative speed of the plane and the wind along the heading axis, is also needed and is a critical factor for helping the autopilot system make operational decisions.
- **Wind speed:** The vector of the wind speed (V_w, O_w) is provided for the RL model, in which V_w is the wind speed and O_w is the angle between the wind speed and the aircraft heading. In this research, we consider only these two components of the wind speed on the horizontal plane (excluding the wind speed in the vertical direction).
- **Control information:** The control information used in this study is the last control command sent to the aircraft, and it consists of the rudder, elevator, aileron and throttle commands, which are denoted A_r , A_e , A_a and A_t , respectively.

- **Deviation from the centerline of the airstrip:** It is necessary to keep the aircraft moving along the centerline of the airstrip during the take-off process, so the deviation from the centerline of the airstrip is input to the autopilot algorithm. To compute the deviation, we establish a coordinate system with the starting point of the airstrip as the origin and transform the position data of the aircraft into this coordinate system. The position of an aircraft in this coordinate system is denoted by the vector $P(x, y)$. $G(a, b)$ is a vector that indicates the direction of the airstrip, and the deviation is defined as

$$D = |\vec{P}| \sin \langle \vec{P}, \vec{G} \rangle = \frac{xb - ya}{\sqrt{a^2 + b^2}}. \quad (4)$$

4.1.2. Preprocessing Data

Because the aerodynamic model of a civil aircraft is quite complicated and nonlinear and artificial neural networks are not accurate for nonlinear fitting tasks. To improve the nonlinear expression of multilayer networks, we innovatively propose a method that uses preprocessing data as inputs to the RL model. The preprocessing scheme processes some flight status data before they are fed into the model using a trigonometric function, an exponential function, an integral function or any other nonlinear mapping as needed. This method can be regarded as computing parts of mappings of the complex flight dynamics in advance and then using the subsequent neural networks to fit the remaining dynamics.

For a civil aircraft with a conventional wing design, the lift force can be described as

$$Y = \frac{1}{2} C_y \rho V^2 S, \quad (5)$$

where C_y is the lift coefficient, ρ is the atmospheric density at the altitude of the aircraft, and S is the area of the wing. Under the condition of a mild turbulent flow, V is approximately equal to the true airspeed V_t . Therefore, we define the preprocessing function of the true airspeed as

$$U(V_t) = V_t^2. \quad (6)$$

Control surfaces such as the rudder, elevator and aileron are mechanisms that control the aircraft's heading and posture by interacting with the surrounding atmosphere. In general, a rotational command from the flight control computer comprises angular data within a specific range; however, as Figure 1 shows, the force between the control surface and the airflow is proportional not to its rotation angle β but to the effective force area:

$$Z = S \times \sin(\beta), \quad (7)$$

where S is the area of the control surface. Therefore, we define the preprocessing function for the 3 control data variables (A_r, A_e, A_a) as

$$U(A_i) = \sin(A_i). \quad (8)$$

Figure 1. The effective force area for the control surface against the airflow.

As Figure 2 shows, the wind speed (V_w, O_w) can be described as a divided, and this symmetric format may be easier for neural networks to understand. Therefore, we provide an additional preprocessed scheme (V_{wx}, V_{wy}) for the wind speed for the learning model:

$$\begin{cases} V_{wx} = V_w \times \sin(O_w) \\ V_{wy} = V_w \times \cos(O_w) \end{cases} \quad (9)$$

Figure 2. A schematic diagram of wind speed decomposition.

4.1.3. Visual Information

A scene obtained by looking out from the main pilot's position implicitly contains not only the motion information but also the spatial information in front of the aircraft; this information is not available from common sensors. As Figure 3 shows, the visual information we use in this paper is composed of images, namely, screenshots from the video stream of the flight simulator. After a screenshot, we resize the image to be uniform.

Figure 3. A scene looking out from the main pilot's position.

Table 1. The state data fed into the reinforcement learning model.

Symbol	Description
P_x	Longitude
P_y	Latitude
P_z	Altitude
R_p	Pitch
R_r	Roll
R_h	Heading
V_x	Velocity in the longitude direction
V_y	Velocity in the latitude direction
V_z	Velocity in the altitude direction
V_p	Rotational velocity of pitch
V_r	Rotational velocity of roll
V_h	Rotational velocity of heading
V_t	True airspeed
V_w	Wind speed
O_w	The angle between wind speed and the aircraft heading
A_r	The last control command on the rudder
A_e	The last control command on the elevator
A_a	The last control command on the aileron
A_t	The last control command on the throttle
D	Deviation from the centerline of airstrip
V_t^2	Preprocessing function for true airspeed
$\sin(A_r)$	Preprocessing function for rudder control
$\sin(A_e)$	Preprocessing function for elevator control
$\sin(A_a)$	Preprocessing function for aileron control
$V_w \times \sin(O_w)$	Preprocessing function for wind speed
$V_w \times \cos(O_w)$	Preprocessing function for wind speed

4.2. Reward Function

In reinforcement learning, a reward function is a regulation that evaluates actions and should be designed according to expert experience. In this paper, to enable the aircraft to move along the airstrip and take off and reach the target area, the reward function is designed to comprise out-of-bounds punishments and rewards for tentative movements.

4.2.1. Out-of-Bounds Punishments

We set boundaries to constrain the aircraft's movement within the expected area B . In Figure 4, these boundaries are marked with red lines. In the horizontal direction, when the aircraft is on the runway, it should keep moving within the two red lines that are at

a distance d_1 from the centerline, and when the aircraft leaves the ground, it should fly within the bounds of the two red dotted lines at a distance d_2 from the centerline. In the vertical direction, at stages L_1 and L_2 , the altitude of the aircraft should not be higher than the red line, and at stage L_3 , the aircraft should fly between the two red lines. If the aircraft flies out of bounds after a tentative action, it will receive a punishment r_p .

Figure 4. The available area for the aircraft when learning to take off. Panel (a) shows the horizontal constraints, and panel (b) shows the constraints in the vertical direction.

4.2.2. Rewards for Tentative Movements

In the learning process, each tentative movement will obtain an immediate reward that represents the value of the attempt. The goal of the autopilot program is to enable the aircraft to overcome crosswind interference, learn to move on the runway, and learn to take off; thus, the closer the aircraft is to the centerline of the runway, the higher the reward. We define this reward relative to the deviation from the centerline as

$$r_d = \partial_d \frac{1}{D^2}, \quad (10)$$

where D is defined at Equation (4) and ∂_d is a scaling factor.

Target point $A(x_a, y_a, z_a)$ is assigned by the experience of a professional pilot. At step t , the vector $P_{t+1}(x_{t+1}, y_{t+1}, z_{t+1})$ denotes the new position of the aircraft, and $P_t(x_t, y_t, z_t)$ denotes the previous position. L_{t+1} denotes the distance between P_{t+1} and the target point, and L_t is the distance between P_t and the target point. The difference between L_t and L_{t+1} is also used as part of the reward and is defined as

$$r_l = \partial_p (L_t - L_{t+1}), \quad (11)$$

where

$$L_t = \sqrt{(x_t - x_a)^2 + (y_t - y_a)^2 + (z_t - z_a)^2} \quad (12)$$

and

$$L_{t+1} = \sqrt{(x_{t+1} - x_a)^2 + (y_{t+1} - y_a)^2 + (z_{t+1} - z_a)^2}. \quad (13)$$

From the above discussion, the reward function is defined as

$$\begin{cases} r = r_d + r_l & : p_{t+1} \in B \\ r = r_p & : p_{t+1} \notin B. \end{cases} \quad (14)$$

4.3. Experience Replay

The DDPG structure on which our method based is an off-policy reinforcement learning method, and experience replay is an important component for off-policy learning [21]. At step t , the transition data that are collected and used to train the model consist of flight status data, visual data, the action, the reward, the next flight status data and the next visual data, and the transition is denoted as $T_t = [S_{s_t}, S_{i_t}, A, R, S_{s_{t+1}}, S_{i_{t+1}}]$.

As Figure 5 shows, with changes in sunlight, the scene from the cockpit will vary over the course of a day. The diurnal cycle of sunlight is 24 h, and the experience memory of reinforcement learning is hardly capable of storing such a large amount of image data. As a result, new information will overwrite previous images, causing an incomplete data distribution in the experiential memory. To address this issue, we create 24 independent memories, where each memory works only in the corresponding hour. At each transition collection, we use the prioritized experience replay method [37] that store transitions based on priorities involved with temporal difference error (TD-Error) [38], and the new transition is stored in a certain memory corresponding to the current hour. When training the autopilot model, we select an equal number of transitions from each memory to constitute a batch.

Figure 5. The change in a scene looking out from the aircraft cockpit over 24 h.

4.4. Architecture of RL Model

In this study, the reinforcement learning algorithm which is used to learn the autopilot function exhibits an actor-critic structure that based on DDPG. As Figure 6 shows, the actor network accepts multimodal data and outputs the next action $A = (A_r, A_e, A_a, A_t)$. The flight status data and preprocessing data are fused through a fully-connected layer. The image data, which are resized to $320 \times 180 \times 3$, are processed by 2 convolutional layers, 2 max-pooling layers and a fully-connected layer. Finally, these feature data are integrated into 2 fully-connected layers.

Figure 6. The architecture of the actor network in the proposed reinforcement learning model.

The critic network has a similar structure to the actor network when processing the image data, and additional action data are offered as input information. The number of units in the last two fully-connected layers is different from that within the actor network. Figure 7 illustrates the structure of the critic network in detail.

4.5. Implementation Details

In this study, we perform experiments with the flight simulator X-Plane, a Boeing 737 model and the research tool—X-Plane connect. The RL program is compiled in Python and Tensorflow, and each trial has 3000 min learning. The X-Plane simulator and the RL program are run on the same computer, and Table 2 lists the interacting data between them. The visual data are continuously collected by taking screenshots of the X-Plane window. At the start of each episode, the aircraft is reset and placed at the starting point of the runway.

Figure 7. The architecture of the critic network in the proposed reinforcement learning model.

Table 2. The interacting data between X-Plane and the reinforcement learning (RL) program.

Data Received From X-Plane	Data Sent to X-Plane
Longitude	Operations on elevator
Latitude	Operations on aileron
Altitude	Operations on rudder
Angle of pitch	Operations on throttle
Angle of roll	
Heading	
Velocity along the longitude	
Velocity along the latitude	
Velocity along the altitude	
Angular rate of change of pitch	
Angular rate of change of roll	
Angular rate of change of heading	
True air speed	

Based on the DDPG algorithm, the actor network and the critic network both have two sets: one is called the evaluation network, and the other is called the target network. The evaluation network containing the current policy is used for learning within the environment and to make action decisions A_t and the critic $Q(S_t, A_t)$. Because the correlation between S_t and S_{t+1} will make the critic $Q(S_{t+1}, A_{t+1})$ inaccurate, the target network containing the policies of the previous few steps is used to give the critic $Q'(S_{t+1}, A_{t+1})$. Then, the TD-Error that used in experience replay can be computed by $E_t = ((r + \gamma Q'(S_{t+1}, A_{t+1})) - Q(S_t, A_t))^2$. Table 3 shows the detailed hyperparameter configurations for the proposed RL algorithm. The Algorithm 1 shows the core steps in pseudocode, the input data of this algorithm is the state information S_t , and the output is the target actor $\mu'(s|\theta^{\mu'})$.

Algorithm 1 Core steps of the proposed RL algorithm

Randomly initialize eval critic network $Q(s, a|\theta^Q)$ and eval actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ .

Initialize target network Q' and μ' with weight $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$

Initialize experience memory M

Initialize the actor replacement counter $C_a = 0$

Initialize the critic replacement counter $C_c = 0$

Initialize the actor replacement interval I_a

Initialize the critic replacement interval I_c

for $episode = 1$ to Z **do**

Initialize a random process $\mathcal{N}$ for action exploration.

Initialize the aircraft and observe the state s_t .

for $step = 1$ to L **do**

Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise.

Run action a_t and compute reward r_t according to the method in Section 3.2.

Observe new state s_{t+1}

Store transition $T_t = (s_t, a_t, r_t, s_{t+1})$ in M .

Sample K transitions (s_i, a_i, r_i, s_{i+1}) from M .

Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$

Update the critic by minimizing the loss: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$

Update the actor policy using the sampled policy gradient (Equation (3)).

if $C_a \% I_a == 0$ **then**

$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$

end if

if $C_c \% I_c == 0$ **then**

$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$

end if

Update C_a and C_c by $C_a = C_a + 1; C_c = C_c + 1$

end for

end for

Get the target actor $\mu'(s|\theta^{\mu'})$.

Table 3. The detailed configuration of the proposed RL algorithm.

Memory capacity (one of the 24 h)	20,000
Learning rate (actor)	9^{-4}
Learning rate (critic)	1^{-3}
Optimization method	Adam
Discount factor γ	0.9
The actor replacement interval I_a	800
The critic replacement interval I_c	600
Soft update factor τ	0.98
Batch size	32

5. Experiments

5.1. Experiment 1: Learning to Take Off under Crosswind Conditions at Different Speeds

In this experiment, we conduct 4 trials under crosswind conditions (10 knots at 90° , 18 knots at 90° , 12 knots at 135° , and 12 knots at 45° , and we compute the average reward by

$$R_a = 0.9 \times R_{a'} + 0.1 \times \sum_{t=1}^N r_t, \quad (15)$$

where, $R_{a'}$ is the last recorded average reward, and N is the total number of steps in one episode.

Results and discussion: Figure 8 shows the learning curves of these trials conducted under crosswind conditions at different speeds. From this figure, we can note that the proposed algorithm can allow a civil aircraft to learn to take off under various crosswinds. In the first 1000 min, the learning performance rise rapidly, and then the learning get into a slow ascent stage accompanied by small shocks. Comparing panel (a) and panel (b), it is obvious that a faster wind speed will make the learning harder and decrease the performance. From panel (c) and panel (d), we can know that the model can learn well both under headwind and tailwind conditions. Figure 9 shows the motion trails of the aircraft when it takes off, and we can find that under different crosswinds, the aircraft can fly within the requested horizontal area and reach the target altitude area at the specified time. Same as the result in Figure 8, the lower the wind speed, the better the flight performance.

Figure 8. The learning curves of the proposed reinforcement learning algorithm under crosswind conditions at different speeds: (a) shows the result of a trial in a 10-knot, 90° crosswind; (b) shows the result of a trial in an 18-knot, 90° crosswind; (c) shows the result of a trial in a 12-knot, 135° crosswind; and (d) shows the result of a trial in a 12-knot, 45° crosswind.

Figure 9. The performance of takeoff under different crosswind conditions after learning. The red line shows the motion trail of the aircraft when it takes off. In each figure, the area between two orange dotted lines is the target area designed based on experience. (a1): the motion trail of the horizontal direction in the 10-knot, 90° crosswind; (a2): the motion trail of the vertical direction in the 10-knot, 90° crosswind; (b1): the motion trail of the horizontal direction in the 18-knot, 90° crosswind; (b2): the motion trail of the vertical direction in the 18-knot, 90° crosswind; (c1): the motion trail of the horizontal direction in the 12-knot, 135° crosswind; (c2): the motion trail of the vertical direction in the 12-knot, 135° crosswind; (d1): the motion trail of the horizontal direction in the 12-knot, 45° crosswind; (d2): the motion trail of the vertical direction in the 12-knot, 45° crosswind.

5.2. Experiment 2: Comparison of Learning with and without Visual Data

In order to observe the implications of visual information, we conduct a trial using our proposed method without supplying visual data. This trial is conducted in a 10-knot, 90° crosswind. The way of computing the average reward is similar to that of Experiment 1.

Results and discussion: Figure 10 clearly shows a comparison of the learning curves from the proposed method with and without visual data. This demonstrates that learning with visual data is more stable and can ultimately acquire a higher score. Though the learning without visual data has a faster learning rate at the initial stage, this advantage fades away after learning for a few hundred minutes. Figure 11 also indicates that the RL model using visual data can make the flight more stable, and the distance to the target point is also closer at the end of the flying.

Figure 10. A comparison of the learning curves from the proposed reinforcement learning algorithm with and without visual data.

Figure 11. A comparison of the takeoff performance of the proposed reinforcement learning algorithm with and without visual data. (a): the compared motion trails of the horizontal direction; (b): the compared motion trails of the vertical direction.

5.3. Experiment 3: Comparison of Learning with and without Preprocessing Data

In this experiment, we conduct a trial using our proposed method without supplying preprocessing data, and compare its learning performance with that of the original proposed method. Similar to Experiment 2, this trial is also conducted in a 10-knot, 90° crosswind, and the same way of computing average reward as that of Experiment 1 is used.

Results and discussion: As Figure 12 shows, compared with the trial without supplying preprocessing data the trial with preprocessing data can achieve better performance in many aspects, including a better learning stability, faster convergence rate and higher final score. This experiment clearly demonstrates the positive influence brought about by supplying preprocessing data. From Figure 13 we can clearly see that the model using preprocessed data can make the flight trail swing less when taking off.

Figure 12. A comparison of the learning curves from the proposed reinforcement learning algorithm with and without preprocessing data.

Figure 13. A comparison of the takeoff performance of the proposed reinforcement learning algorithm with and without preprocessing data. (a): the compared motion trails of the horizontal direction; (b): the compared motion trails of the vertical direction.

6. Conclusions and Future Studies

In this work, we proposed a reinforcement learning method to learn to accomplish a challenging task—learning to have a civil aircraft take off under crosswind conditions. Our method feeds common flight data, visual data and preprocessing data into the RL model. Experiments under different crosswind conditions demonstrated that the proposed method can effectively accomplish this learning task. Additionally, further comparative experiments indicated the advantages of supplying multimodal data and preprocessing data in the learning. Compared with traditional autopilot algorithms, the proposed algorithm can complete more complex autopilot tasks, and it can be easily applied to other autonomous flying tasks. Using unsupervised machine learning methods, it can reduce a lot of manual modeling work that differ from task to task.

A drawback of our method that is worthy of discussion is that the use of multimodal data and preprocessing data will complicate the architecture of the RL network and use more nerve units. As a result, more GPU memory and more learning hours will be required. However, this issue will become increasingly negligible with the development of computer hardware year after year. The main purpose of this research is to study whether machine learning is capable of the autopilot of aircraft under abnormal weather conditions, however the stability of the RL model cannot be checked, so, considering security, the proposed method can only work on a flight simulator, and it can be used to train pilots in a flight simulation environment. In subsequent studies, we plan to make machine learning and traditional control algorithms work together so that can make full use of the intelligent advantages of machine learning and the stable advantages of traditional control algorithms for improving the ability of autopilot of aircrafts. In addition, we will use machine learning to try to accomplish more difficult autonomous flight tasks, such as implementing the autopilot function under wind shear conditions and learning to have an aircraft recover from stalling.

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

Funding: This work was supported by Major Project of the New Generation of Artificial Intelligence, China (No. 2018AAA0102900).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

- Royo, P.; Pablo, E. Autopilot abstraction and standardization for seamless integration of unmanned aircraft system applications. *J. Aerosp. Comput. Inf. Commun.* **2011**, *8*, 197–223. [CrossRef]
- Theis, J.; Daniel O.; Frank T.; Harald P. Robust autopilot design for landing a large civil aircraft in crosswind. *Control Eng. Pract.* **2018**, *76*, 54–64. [CrossRef]
- Dehais, F.; Peysakhovich, V.; Scannella, S.; Fongue, J.; Gateau, T. “Automation Surprise” in Aviation: Real-Time Solutions. In Proceedings of the 33rd annual ACM conference on Human Factors in Computing Systems, Seoul, Korea, 18–23 April 2015; pp. 2525–2534.
- Cooling, J.E.; Herbers, P.V. Considerations in autopilot litigation. *J. Air L. Com.* **1982**, *48*, 693.
- Sharma, V.; Voulgaris, P.G.; Frazzoli, E. Aircraft autopilot analysis and envelope protection for operation under icing conditions. *J. Guid. Control Dyn.* **2004**, *27*, 454–465. [CrossRef]
- Patsko, V.S.; Botkin, N.D.; Kein, V.M.; Turova, V.L.; Zarkh, M.A. Control of an aircraft landing in windshear. *J. Optim. Theor. Appl.* **1994**, *83*, 237–267. [CrossRef]
- Botkin, N.; Turova, V.; Diepolder, J.; Bittner, M.; Holzapfel, F. Aircraft control during cruise flight in windshear conditions: Viability approach. *Dyn. Games Appl.* **2017**, *7*, 594–608. [CrossRef]
- Blakelock, J.H. *Automatic Control of Aircraft and Missiles*; John Wiley & Sons, Inc.: New York, NY, USA, 1991; pp. 10–53.
- Sanil, N.; Rakesh, V.; Mallapur, R.; Ahmed, M.R. Deep Learning Techniques for Obstacle Detection and Avoidance in Driverless Cars. In Proceedings of the 2020 International Conference on Artificial Intelligence and Signal Processing (AISP), Amaravati, India, 10–12 January 2020; pp. 1–4.
- Hodges, C.; An, S.; Rahmani, H.; Bennamoun, M. Deep learning for driverless vehicles. In *Handbook of Deep Learning Applications*; Springer: Cham, Switzerland, 2019; pp. 83–99.
- Han, M.; Senellart, P.; Bressan, S.; Wu, H. Routing an autonomous taxi with reinforcement learning. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, Indianapolis, Indiana, 24–28 October 2016; pp. 2421–2424.
- Kendall, A.; Hawke, J.; Janz, D.; Mazur, P.; Reda, D.; Allen, J.M.; Shah, A. Learning to drive in a day. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 8248–8254.
- Heim, E.H.; Viken, E.; Brandon, J.M.; Croom, M.A. NASA’s Learn-to-Fly Project Overview. In Proceedings of the 2018 Atmospheric Flight Mechanics Conference, Atlanta, Georgia, 25–29 June 2018; p. 3307.
- Anwar, M.A.; Raychowdhury, A. NavREN-RL: Learning to fly in real environment via end-to-end deep reinforcement learning using monocular images. In Proceedings of the 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), Stuttgart, Germany, 20–22 November 2018; pp. 1–6.
- Shaker, M.; Smith, M.N.; Yue, S.; Duckett, T. Vision-based landing of a simulated unmanned aerial vehicle with fast reinforcement learning. In Proceedings of the 2010 International Conference on Emerging Security Technologies, Canterbury, UK, 6–7 September 2010; pp. 183–188.
- Caruana, R.; Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 161–168.
- Cunningham, P.; Cord, M.; Delany, S.J. Supervised learning. In *Machine Learning Techniques for Multimedia*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 21–49.
- Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018; pp. 5–23.
- Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. *arXiv* **2015**, arXiv:1509.06461. Available online: <https://arxiv.org/pdf/1509.06461> (accessed on 12 June 2020).
- Koch, W.; Mancuso, R.; West, R.; Bestavros, A. Reinforcement learning for UAV attitude control. *ACM Trans. Cyber-Phys. Syst.* **2019**, *3*, 1–21. [CrossRef]
- Lillicrap, T.P.; Hunt, J.J.; Pritzel, A. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971. Available online: <https://arxiv.org/pdf/1509.02971> (accessed on 21 June 2020).
- Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the 32nd international conference on machine learning, Lille, France, 7–9 June 2015; pp. 1889–1897.
- Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347. Available online: <https://arxiv.org/pdf/1707.06347> (accessed on 10 July 2020).
- Xie, J.; Peng, X.; Wang, H.; Niu, W.; Zheng, X. UAV Autonomous Tracking and Landing Based on Deep Reinforcement Learning Strategy. *Sensors* **2020**, *20*, 5630. [CrossRef] [PubMed]
- Dynkin, E.B. Markov processes. In *Markov Processes*; Springer: Berlin/Heidelberg, Germany, 1965; pp. 77–104.
- Bellman, R. Dynamic programming and stochastic control processes. *Inf. Control* **1958**, *1*, 228–239. [CrossRef]
- Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
- Mnih, V.; Kavukcuoglu, K.; Silver, D. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602. Available online: <https://arxiv.org/pdf/1312.5602> (accessed on 6 July 2020).

-
29. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M. Deep reinforcement learning: A brief survey. *IEEE Signal Process Mag.* **2017**, *34*, 26–38. [[CrossRef](#)]
 30. Wei, F.; Amaya-Bower, L.; Gates, A.; Rose, D.; Vasko, T. The Full-Scale Helicopter Flight Simulator Design and Fabrication at CCSU. In Proceedings of the 57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, San Diego, CA, USA, 4–8 January 2016; p. 0582.
 31. Jirgl, M.; Boril, J.; Jalovecky, R. The identification possibilities of the measured parameters of an aircraft model and pilot behavior model on the flight simulator. In Proceedings of the International Conference on Military Technologies (ICMT), Brno, Czech Republic, 19–21 May 2015; pp. 1–5.
 32. Kaviyarasu, A.; Senthil Kumar, K. Simulation of Flapping-wing Unmanned Aerial Vehicle using X-plane and Matlab/Simulink. *Defence Sci. J.* **2014**, *64*, 327–331. [[CrossRef](#)]
 33. Baomar, H.; Bentley, P.J. An Intelligent Autopilot System that learns piloting skills from human pilots by imitation. In Proceedings of the 2016 International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA, 7–10 June 2016; pp. 1023–1031.
 34. Teubert, C.; Watkins, J. The X-Plane Connect Toolbox (2020). Available online: <https://github.com/nasa/XPlaneConnect> (accessed on 2 August 2020).
 35. Wayllace, C.; Ha, S.; Han, Y.; Hu, J.; Monadjemi, S.; Yeoh, W.; Ottley, A. DRAGON-V: Detection and Recognition of Airplane Goals with Navigational Visualization. In *AAAI 2020*; AAAI: Menlo Park, CA, USA, 2020; pp. 13642–13643. [[CrossRef](#)]
 36. Fremont, D.J.; Chiu, J.; Margineantu, D.D.; Osipychev, D.; Seshia, S.A. Formal Analysis and Redesign of a Neural Network-Based Aircraft Taxiing System with VerifAI. *arXiv* **2005**, arXiv:2005.07173. Available online: <https://arxiv.org/pdf/2005.07173> (accessed on 5 August 2020).
 37. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952. Available online: <https://arxiv.org/pdf/1511.05952> (accessed on 28 August 2020).
 38. Tesauro, G. Temporal difference learning and TD-Gammon. *Commun. ACM* **1995**, *38*, 58–68. [[CrossRef](#)]