drones

MDPI

Review

A Survey of Offline- and Online-Learning-Based Algorithms for Multirotor Uavs

Serhat Sönmez *, Matthew J. Rutherford and Kimon P. Valavanis

D. F. Ritchie School of Engineering and Computer Science, University of Denver, Denver, CO 80210, USA; matthew.rutherford@du.edu (M.J.R.); kimon.valavanis@du.edu (K.P.V.)

* Correspondence: serhat.sonmez@du.edu

Abstract: Multirotor UAVs are used for a wide spectrum of civilian and public domain applications. Their navigation controllers include onboard sensor suites that facilitate safe, autonomous or semi-autonomous multirotor flight, operation, and functionality under nominal and detrimental conditions and external disturbances, even when flying in uncertain and dynamically changing environments. During the last decade, given the available computational power, different learning-based algorithms have been derived, implemented, and tested to navigate and control, among other systems, multirotor UAVs. Learning algorithms have been and are used to derive data-driven based models, to identify parameters, to track objects, to develop navigation controllers, and to learn the environments in which multirotors operate. Learning algorithms combined with model-based control techniques have proven beneficial when applied to multirotors. This survey summarizes the research published since 2015, dividing algorithms, techniques, and methodologies into offline and online learning categories and then further classifying them into machine learning, deep learning, and reinforcement learning sub-categories. An integral part and focus of this survey is on online learning algorithms as applied to multirotors, with the aim to register the type of learning techniques that are either hard or almost hard real-time implementable, as well as to understand what information is learned, why, how, and how fast. The outcome of the survey offers a clear understanding of the recent state of the art and of the type and kind of learning-based algorithms that may be implemented, tested, and executed in real time.

Keywords: multirotor UAVs; offline learning; online learning; reinforcement learning; deep learning; machine learning

check for updates

Citation: Sönmez, S.; Rutherford, M.J.; Valavanis, K.P. A Survey of Offline- and Online-Learning-Based Algorithms for Multirotor UAVs. *Drones* **2024**, *8*, 116. https://doi.org/10.3390/drones8040116

Academic Editor: Abdessattar Abdelkefi

Received: 1 February 2024
Revised: 11 March 2024
Accepted: 20 March 2024
Published: 22 March 2024

1. Introduction

Unmanned aerial vehicles (UAVs) have witnessed unprecedented levels of growth during the last 20 years, with civilian and public domain applications spanning power line inspection [1], the monitoring of mining areas [2], wildlife monitoring and conservation [3], border protection [4], building and infrastructure inspection [5], and precision agriculture [6], to name but a few applications. Although different UAV types and configurations have been utilized for such applications, multirotor UAVs, particularly quadrotors, are the most commonly and widely used due to their perceived advantages, effectiveness, hovering capabilities, and efficiency during flight.

A plethora of conventional and advanced model-based linear, linearized, and nonlinear controllers have already been derived and used for multirotor navigation and control. However, recently, learning-based algorithms and techniques have gained momentum because they facilitate and allow, among other things, for (i) data-driven system modeling that may also include model updates; (ii) combined data-driven and model-based modeling and control and parameter identification; (iii) data-driven parameter identification; (iv) data-driven environment model development; and (v) pure learning-based control.

Stated differently, learning-based approaches add to the model-based formulation, enhance multirotor modeling and control, and offer alternatives for learning, modeling, and understanding the environment.

Learning-based algorithms are basically divided into offline and online learning, although there exist some learning algorithms that include both offline and online learning components. Most researchers have extensively studied offline-learning-based algorithms and applied them to different families of (linear and nonlinear) systems. The derivation and implementation of online-learning-based algorithms in multirotor UAVs is a relatively recent area of research that has attracted increased interest because of the real-time implementability potential, which may facilitate continuous anytime online learning. This momentum has motivated the present survey.

To begin with, a review of the literature reveals that there exists considerable published research addressing the use of learning algorithms for UAV control. The emphasis of already-published surveys is on developing and adopting machine learning, deep learning, or reinforcement learning algorithms. To be specific, Carrio et al. [7] focused on deep learning methods that are applied to UAVs, while in Polydoros and Nalpanditis [8], the emphasis was on model-based reinforcement learning techniques that are applied to robotics but also have applications to UAVs. Machine learning algorithms for UAV autonomous control were explored by Choi and Cha [9], while Azar et al. [10] investigated deep reinforcement learning algorithms as applied to drones. Most recently, Brunke et al. [11] presented several learning algorithm-based applications in robotics, including multirotor UAVs.

The common theme of already-published surveys is that they discuss different offline-learning-based control algorithms that may be, or have been, applied to different UAV types, but they are not real-time implementable. Therefore, in contrast to the existing surveys, the focus of this research is on also registering the online-learning-based algorithms that have shown potential and/or have been implemented and tested on multirotor UAVs.

Without loss of generality, for the purpose of this survey, the below-provided 'attributes' are considered important, and they facilitated the review process. The list is by no means complete, nor unique; it may be modified and enhanced accordingly based on set objectives. Note that in what follows, the terms 'agent' and 'multirotor' are used interchangeably:

1. Navigation task: This refers to the (autonomous or semi-autonomous) function that the multirotor needs to accomplish, given a specific controller design and/or configuration.
2. Learning: This refers to 'what' the agent learns in order to complete the navigation task.
3. Learning algorithm: This refers to the specific algorithm that needs to be followed for the agent to learn. Inherent in this attribute is 'what' is being learned by the agent, and 'how'.
4. Real-time applicability: This refers to 'how fast' learning is achieved and 'how computationally expensive' the learning algorithm is, which basically dictates whether learning is applicable in hard real time or in almost hard real time. Stated differently, the answer to 'how fast' determines the implementability of the learning algorithm. The calculation of the algorithm's computational complexity may also provide additional information on 'how fast' the agent learns.
5. Pros and Cons: This refers to the advantages and limitations of the underlying learning approach, which, in unison with all other attributes, determines the overall applicability and implementability of the learning approach on multirotor UAVs.

The rest of the survey is organized as follows: Section 2 provides background information and related definitions, which are deemed essential for clarification and classification purposes. Section 3 summarizes offline learning and provides a detailed table reflecting published research, also stating what is being learned. The review of offline learning techniques is essential for completeness purposes and also for an understanding of the differences between offline and online learning. Section 4 dives into online learning approaches. An overview of each learning method is presented, along with what is being learned and why, the advantages and disadvantages, and the obtained results. The discussion and conclusions are offered in Section 5.

2. Background Information

Key concepts and definitions are presented next. The related and relevant literature is cited to support statements, findings, and observations. This information is adopted and used throughout the paper; it also helps to correctly classify the reviewed learning-based algorithms, when needed.

Definitions

Reinforcement Learning: Reinforcement learning, RL, is a machine learning technique in which an agent communicates with the environment to find the best action using the state space and a reward function. RL includes four main components: a policy, a reward signal, a value function, and, optionally, an environment model [7,12]. RL may be either online or offline. The general configurations of offline RL and online RL approaches are shown in Figures 1 and 2.

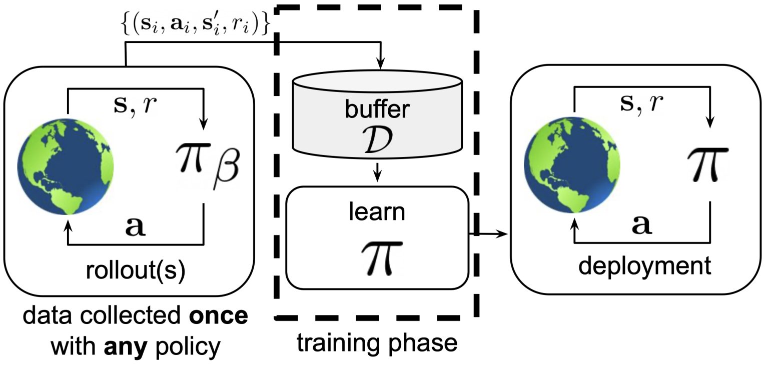

Figure 1. Offline reinforcement learning block diagram illustration [13].

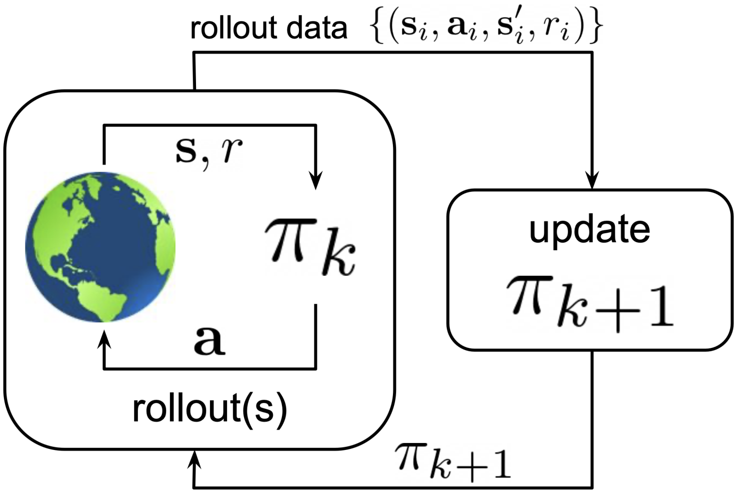

Figure 2. Online reinforcement learning block diagram illustration [13].

Policy: In the context of learning algorithms, a policy determines how the learning agent behaves at a given moment. A policy maps the discerned states of the environment to the actions that the agent should take when in those states [12]. The policy may reflect either on-policy or off-policy options.

On-policy: On-policy techniques relate to a policy that is used to make decisions. Policy iteration, value iteration, Monte Carlo for On-Policy, and State–Action–Reward–State–Action (SARSA) algorithms are representative examples of on-policy algorithms [12].

Off-policy: Off-policy methods refer to a policy that is dissimilar to that used to produce data. Q-learning and deep deterministic policy gradient (DDPG) are examples of off-policy algorithms [12]; see Figure 3 for the configuration of the off-policy RL algorithm. The agent experience is the input to a data buffer $\mathcal{D}$, which is also called a replay buffer. Each new policy π_{k+1} is trained by utilizing the samples of all previous policies, π_0, π_1, …, π_k, that are stored in $\mathcal{D}$.

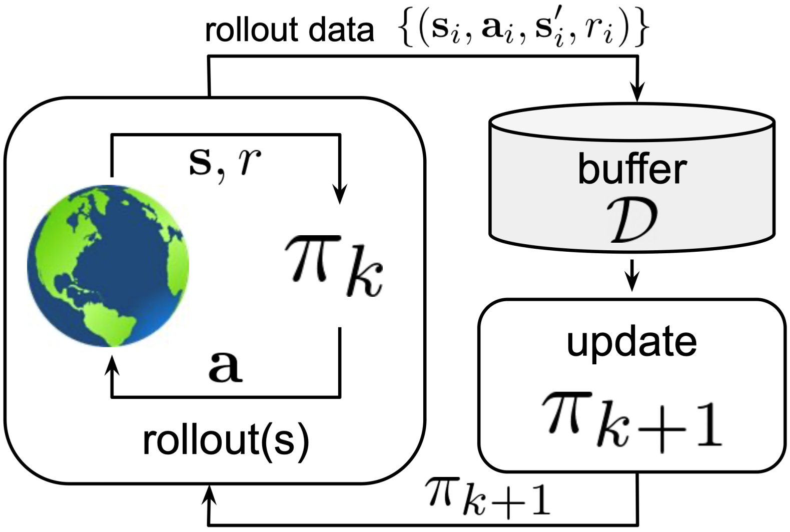

Figure 3. Off-policy RL algorithm configuration [13].

On-policy methods are generally less complicated than off-policy ones, and they are contemplated first. Since data are produced by a different policy in off-policy methods, they converge more slowly. Regardless, off-policy methods provide more powerful and general alternatives [12].

Reward Signal: A reward signal is a target in an RL algorithm. The agent receives a single number, the reward, from the environment at each time step. The only objective of the agent is to maximize the total reward over time. The reward signal provides an immediate sense of the current state and indicates what events are beneficial or detrimental to the agent [12].

Value Function: A value function determines what events are advantageous to the agent in the long term.

Environment Model: An environment model helps mimic and replicate the behavior of the environment. It allows for making inferences about how the environment will behave.

In what follows, the definitions and fundamental concepts of offline and online learning, supervised and unsupervised learning, and machine and deep learning are detailed.

Offline Learning: In offline learning, the learning algorithm trains an agent or artificial neural network (ANN). The agent interacts with the environment, and it is updated during the training process. However, the agent is never updated after training is completed. Bartak and Vykovsky [14] and Edhah et al. [15] present representative examples of offline machine learning (ML) and deep learning (DL) algorithms, respectively. The studies of Xu et al. [16], Rodriguez et al. [17], and Yoo et al. [18] are representative examples of offline RL algorithms. As observed in the configuration diagram of offline RL in Figure 1, a dataset ($\mathcal{D}$) is collected by the behavior policy (π_β) with the help of the states (s) and the reward function (r). A policy (π) is trained by using $\mathcal{D}$. The training process does not interact with the Markov decision process (MDP). The trained policy (π) is deployed to control the system. The policy (π) interacts with the environment using the states (s) and the reward function (r) and creates the action space (a).

Online Learning: In online learning, according to Hoi et al. [19], the learner keeps on learning and improves prediction to reach the best possible one by interacting with the environment, as shown in Figure 2. A policy (π_k) creates the action space to interact with the environment using the states (s) and the reward function (r). Then, π_k is updated by using the rollout data, including states, actions, future states, and the reward function. After the updated policy (π_{k+1}) is determined, it is replaced with the current policy (π_k).

Note that for the purposes of this survey, the definition of online learning is extended to account for cases where the agent continues the learning process during operation in real time, even after completing the offline learning process or without any basic learning. This extension allows for 'anytime learning' while the underlying system continues to function and accounts for the ability to modify the system model and also adapt its parameter values (time-varying systems).

Supervised Learning: In supervised learning, the learning algorithm learns from a classified and labeled dataset [7].

Unsupervised Learning: In unsupervised learning, the learning algorithm utilizes unlabeled data, which are collected from sensors, to learn the proposed task. Unsupervised learning techniques are widely used in RL [7].

Machine Learning: Machine learning, ML, is a component of Artificial Intelligence (AI), in which tasks are learned (or imitations of tasks are learned) from collected data [7].

Deep Learning: Deep learning, DL, is a category or subset of ML that involves the use of deep neural networks, DNNs, with input, hidden, and output layers to model and solve complex problems.

Moreover, RL techniques and methods are divided into model-based and model-free ones.

Model-based: In a model-based method, the agent predicts the future states and reward and also chooses the action that provides the highest expected reward [12]. Models and planning are used to solve RL problems.

Model-free: In a model-free method, the agent does not utilize the environment model but makes decisions by only using trial-and-error approaches [12].

The main difference between model-based and model-free methods is that the former relies on planning, while the latter relies on learning [12].

A Google Scholar search for articles published since 2015 returns the paper distribution that is illustrated in Figure 4, which shows the number of published offline and online learning papers that deal with multirotor UAVs.

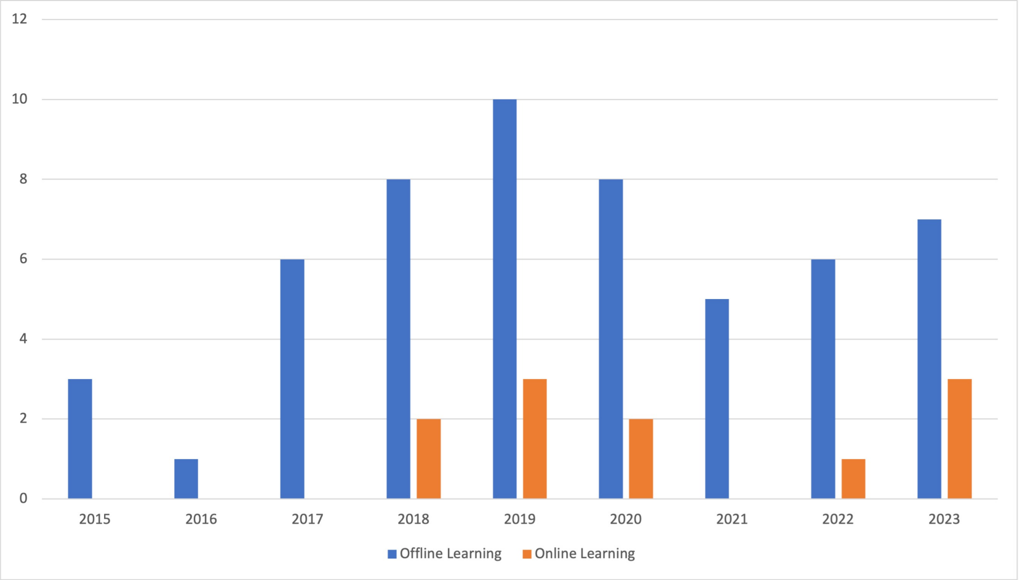

Figure 4. Publications on online and offline learning algorithms for control of multirotor UAVs since 2015 based on Google Scholar search.

This section has provided and explained the key definitions that relate to learning algorithms discussed in this survey. The next section reviews offline learning techniques.

3. Offline Learning

In offline learning, the system may be trained either by using collected and/or provided data (supervised learning) or, alternatively, by using feedback before its actual operation, without using any data (unsupervised learning). In this case, when operating in real time, the agent, or the neural network (NN), is not updated or affected by the environment.

Table 1 summarizes offline learning and RL approaches that have been applied to multirotor UAVs. The table includes the publication year of the paper, the adopted or derived learning model, the application task/domain, and what is being learned. Fifty-five articles have been reviewed and classified as machine learning, ML, or deep learning, DL, or reinforcement learning, RL, approaches (as indicated by the authors). In cases where no information is provided in the reviewed papers, the classification follows the provided definitions in Section 2.

Table 1. Offline learning papers.

Year	Authors	Learning Model	Application Task	What Is Being Learned
2015	Bartak et al. [14]	ML	Object tracking	How to detect an object
2015	Giusti et al. [20]	ML	Navigation	Image classification to determine the direction
2018	Kaufmann et al. [21]	ML	Waypoints and the desired velocity	How to detect an object
2021	Janousek et al. [22]	ML	Landing and flight planning	How to recognize an object
2023	Vladov et al. [23]	ML	Stabilization	How to adjust controller parameters
2015	Kim et al. [24]	DL	Navigation	Image classification to assist in flights
2017	Li et al. [25]	DL	Trajectory tracking	Control signals
2017	Smolyanskiy et al. [26]	DL	Navigation	The view orientation and lateral offset
2018	Jung et al. [27]	DL	Navigation	How to detect the center of a gate
2018	Loquercio et al. [28]	DL	Navigation	How to adjust the yaw angle and the probability of collision
2019	Edhah et al. [15]	DL	Hovering	How to determine propeller speed
2019	Mantegazza et al. [29]	DL	Ground target tracking	Image classification for control
2023	Cardenas et al. [30]	DL	Position control	How to determine the rotor speeds
2016	Imanberdiyev et al. [31]	RL	Navigation	How to select the moving direction
2017	Polvara et al. [32]	RL	Landing	How to detect a landmark and control vertical descent
2017	Choi et al. [33]	RL	Trajectory tracking	The control input
2017	Kahn et al. [34]	RL	Avoiding failure	The policy
2017	Hwangbo et al. [35]	RL	Stabilization	How to determine the rotor thrusts
2018	Xu et al. [16]	RL	Landing	How to determine the velocities of the UAV
2018	Lee et al. [36]	RL	Landing	How to determine the roll and pitch angles
2018	Vankadari et al. [37]	RL	Landing	How to determine the velocities of the UAV on the x- and y-axes
2018	Kersandt et al. [38]	RL	Navigation	How to select three actions: move forward, turn right, and turn left
2018	Pham et al. [39]	RL	Navigation	How to select the moving direction
2019	Rodriguez et al. [17]	RL	Landing	How to determine the velocities of the UAV on the x- and y-axes
2019	Liu et al. [40]	RL	Formation control	The optimal control law
2019	Lambert et al. [41]	RL	Hovering	The mean and variance of the changes in states

Table 1. *Cont.*

Year	Authors	Learning Model	Application Task	What Is Being Learned
2019	Manukyan et al. [42]	RL	Hovering	How to determine the rotor speeds
2019	Srivastava et al. [43]	RL	Target tracking	How to determine the velocities of the UAV on the x-, y-, and z-axes
2019	Wu et al. [44]	RL	Trajectory planning	How to select the moving direction
2019	Wang et al. [45]	RL	Navigation	How to determine the steering angle
2019	Zeng and Xu [46]	RL	Path Planning	How to select the flight direction
2020	Yoo et al. [18]	RL	Trajectory tracking	How to adjust PD and LQR controller gains
2020	Rubi et al. [47]	RL	Trajectory tracking	How to determine the yaw angle
2020	Pi et al. [48]	RL	Trajectory tracking	How to determine the rotor thrusts
2020	Zhao et al. [49]	RL	Formation control	How to solve the Bellman equation
2020	Guerra et al. [50]	RL	Trajectory optimization	The control signal
2020	Li et al. [51]	RL	Target tracking	How to determine the angular velocity of the yaw angle and linear acceleration
2020	Kulkarni et al. [52]	RL	Navigation	How to select the moving direction
2020	Hu and Wang [53]	RL	Speed optimization	How to determine the rotor speeds
2021	Kooi et al. [54]	RL	Landing	How to determine the total thrust and the roll and pitch angles
2021	Rubi et al. [55]	RL	Trajectory tracking	How to determine the yaw angle
2021	Bhan et al. [56]	RL	Avoiding failure	How to adjust the gains of the PD position controller
2021	Li et al. [57]	RL	Trajectory planning	How to obtain the parameter vector of the approximate value function
2022	Jiang et al. [58]	RL	Landing	How to determine the velocity of the UAV on the x- and y-axes
2022	Abo et al. [59]	RL	Landing	How to determine the roll, pitch, and yaw angles and the velocity of the UAV on the z-axis
2022	Panetsos et al. [60]	RL	Payload transportation	How to obtain the reference Euler angles and velocity on the z-axis
2022	Ye et al. [61]	RL	Navigation	How to select the moving direction and determine the velocity
2022	Wang and Ye [62]	RL	Trajectory tracking	How to determine the pitch and roll torques
2022	Farsi and Liu [63]	RL	Hovering	How to determine the rotor speeds
2023	Xia et al. [64]	RL	Landing	How to obtain the force and torque command
2023	Ma et al. [65]	RL	Trajectory tracking	How to determine the rotor speeds
2023	Castro et al. [66]	RL	Path Planning	How to find optimized routes for navigation
2023	Mitakidis et al. [67]	RL	Target tracking	How to obtain the roll, pitch, and yaw actions
2023	Shurrab et al. [68]	RL	Target localization	How to determine the linear velocity and yaw angle

3.1. Machine Learning

Most offline ML techniques applied to and used for multirotor UAVs consider an onboard monocular camera. ML-based approaches have been developed and adopted for navigation purposes, for stabilization, to track an object, to pass through waypoints with a desired velocity, and for landing purposes on stationary or dynamic targets. In addition, ML approaches are also used to tune and adjust controller parameters.

In general, captured and acquired images are sent to pre-trained NNs, which first classify the obtained images into different classes and then pass this information to the underlying multirotor controller, as discussed in Bartak and Vykovsky [14], Janousek et al. [22], Giusti et al. [20], and Kaufmann et al. [21]. The specifics are offered next.

Bartak and Vykovsky [14] combined computer vision, ML, and control theory techniques to develop a software tool for a UAV to track an object; the object is selected by a user who observes a series of video frames (images) and picks a specific object to be tracked by the multirotor. *P-N learning*, where *P* and *N* represent positive and negative learning, respectively, is used by a Tracking–Learning–Detection (TLD) algorithm. The Lucas–Kanade tracker is implemented in the tracking phase, and a cascaded classification algorithm that includes an ML technique helps detect the object. A cascaded classification algorithm that consists of a patch variance classifier, an ensemble classifier, and a nearest-neighbor classifier is also used. A simple RL algorithm decides the forward or backward speed of the multirotor by using the scaled size of the object. The yaw angle of the multirotor, used to follow the object, is provided to a Proportional–Integral–Derivative (PID) controller as input to determine the flight direction.

Janousek et al. [22] developed a method to accurately guide an autonomous UAV to land in a specific area, which is labeled as the 'ground object'. The landing area includes a QR code, which, after it is identified and recognized, provides specific instructions/commands to the UAV for landing. An NN is used to identify the landing area using the onboard UAV camera. The recognition process is carried out from a ground control station, GCS, which is a part of the overall UAV-GCS ensemble. The GCS includes a communication channel for command transmission (to and from the UAV). False detection of the landing area may occur (i.e., due to sunlight), and thus, success depends on how accurately the processed image determines the landing area and not on the learning process itself. Regardless, when the UAV is within an 'acceptable flight altitude', a successful landing is accomplished.

Giusti et al. [20] used a quadrotor with an onboard monocular camera to determine the path configuration and direction of forest or mountain trails. A single image is collected from the onboard camera. A DNN is trained using supervised learning to classify the obtained images. Two parameters are defined: $\vec{v}$ for the direction of the camera's optical axis and $\vec{t}$ for the direction of the trail. Based on the calculated angle α between $\vec{v}$ and $\vec{t}$ and the angle β, which is 15° around $\vec{t}$, three actions are determined and classified as *Turn Left (TL)*, *Go Straight (GS)*, and *Turn Right (TR)*, represented by

– TL if $-90^\circ < \alpha < -\beta$;
– GS if $-\beta \leq \alpha < +\beta$;
– TR if $+\beta \leq \alpha < +90^\circ$.

These choices consider that the onboard camera centers and focuses on the motion direction. For performance evaluation and comparison, three alternatives were considered: learning using a Saliency-based model, learning following the method discussed in Santana et al. [69], and learning by using two human observers, who were asked to make one of the three previously mentioned decisions. The accuracy of the DNN is 85.2%; the accuracy of the Saliency-based model is 52.3%; and the accuracy of the model in [69] is 36.5%. The accuracy of Human1 is slightly better than the accuracy of the DNN, 86.5%; Human2 has 82% accuracy, which is lower than the accuracy of the DNN. This methodology has been tested experimentally and has produced successful results.

Kaufmann et al. [21] focused on the problem of autonomous, vision-based drone racing in dynamic environments, with an emphasis on path planning and control. A convolutional neural network (CNN) is used to detect the location of the waypoints from raw images and to decide on the speed to pass through the gates. The planner utilizes this information to design a short minimum-jerk trajectory to reach the targeted waypoints. This technique was tested via simulations and in real environments. Comparisons with other navigation approaches and professional human drone pilots were made. It is shown that this method completes the track slower than human pilots do, but with a higher success rate. The success rate is also much higher compared to using visual-inertial odometry (VIO).

Vladov et al. [23] studied the UAV stabilization problem. They adjust the PID controller parameters using a recurrent multilayer perceptron (RMLP) method to stabilize the UAV attitude angles. The determined error is the input to an NN. Instead of using a constant training rate, an adaptive training rate is implemented to overcome slow convergence in the learning part and to avoid becoming trapped in a local minimum during the learning phase. The results show that this method has a lower attitude error when compared to the RMLP method with a constant training rate and when using only an ANN.

To conclude, in this part, several ML techniques applied to multirotor UAVs have been presented. The emphasis is on the utilization of monocular cameras for waypoint navigation, stabilization, object tracking, landing approaches, and controller parameter tuning. In detail, the reviewed algorithms are centered on object tracking using a combination of computer vision and machine learning; autonomous landing using artificial neural networks; trail path determination using deep neural networks; vision-based drone racing using convolutional neural networks; and UAV stabilization using recurrent multilayer perceptrons with adaptive training rates.

3.2. Deep Learning

DL algorithms (discussed in eight papers) that have been applied to multirotor UAVs focus on navigation and control, hovering, ground target tracking, and trajectory tracking.

In Edhah et al. [15], a DNN was used to control UAV altitude and hover. The standard feedforward, greedy layer-wise, and Long Short-Term Memory (LSTM) methods were evaluated and compared. The controller outputs, position and speed errors, are collected every 1 ms and then used by a supervised learning technique to train the DNN. After training, the trained DNN controller (and related parameters) is replaced by a Linear–Quadratic Regulator (LQR) controller. To overcome a slight offset in the output signal that results in a small error (in the system response), a proportional corrector is added in parallel to the DNN controller to recover the error in the DNN output signal. The best results are obtained when using the greedy layer-wise method.

Mantegazza et al. [29] presented three different approaches and models to track a moving user. In the first model, the ResNet [70] architecture (a CNN) is utilized [28]. Red–green–blue (RGB) images captured from 14 different people are used as input, providing x-, y-, and z-positions as output. The second model follows the same structure, but velocities on the x- and y-axes are provided as additional inputs. These additional inputs skip ResNet, but they are concatenated to the output of the NN. The outputs are control variables corresponding to four moving directions: up, down, left, and right. In the third model, a simple multilayer perceptron is utilized to map the quadrotor position on three axes and velocities on the x- and y-axes to control variables. The Mean Absolute Error (MAE) approach is deployed as a loss function during the training of all three models. The first and second models use a simple baseline controller function. The last approach uses a combination of the first and third models.

Li et al. [25] studied trajectory tracking without any adaptation, but they considered quadrotor stabilization and robustness in the presence of disturbances. A DNN is trained with labeled training examples using a standard feedforward method. The DNN uses the desired quadrotor trajectory and the current states of position and translational velocities (on each axis), Euler angles, angular velocities, and acceleration on the z-axis. The quadrotor reference states are provided as the DNN output. The trained DNN is placed in front of a controller, and errors between current and desired states are used as inputs to a PID controller. The results show that the DNN with current state feedback is more efficient than the DNN without current state feedback. However, the DNN using future desired state feedback provides better performance.

Kim and Chen [24], Jung et al. [27], Loquercio et al. [28], and Smolyanskiy et al. [26] focused on the quadrotor navigation task using DL techniques.

Kim and Chen [24] developed an autonomous indoor navigation system for a quadrotor to find a specific item using a single onboard camera. Six flight commands are used: Move Forward, Move Right, Move Left, Spin Right, Spin Left, and Stop. To establish a dataset, an expert pilot flies the quadrotor in seven different locations; from the UAV, images are collected that are based on or correspond to (specific) flight commands. A

CNN that is a modified CaffeNet model [71] is trained with the established dataset. This modification allows for faster training. During indoor flights, obtained images are classified by the trained NN, and based on image classification, specific control commands are issued to the quadrotor.

Jung et al. [27] developed a CNN to accurately identify the centers of gates during indoor drone racing. ADRNet was built and trained using the Caffe library. To build the ADRNet, the AlexNet used in [72] was applied instead of the VGG-16 employed in [73]. In [74], a convolutional layer was added among the fully connected layers of AlexNet instead of the fc6 and fc7 layers, while the fc8 layer was removed. Thus, a shorter inference time is required compared to the VGG-16-based Single-Shot Detection (SSD) approach. ADRNet detects the center of the gate, and this information is forwarded to a Line-Of-Sight (LOS) guidance algorithm that issues specific flight control commands. The performance of three Single-Shot Detection (SSD) models, the VGG-16-based SSD, the AlexNet-based SSD, and the ADRNet, were compared. ADRNet was the fastest model to detect the center of the gate.

Different from the traditional map–localize–plan methods, Loquercio et al. [28] applied a data-driven approach to overcome the UAV challenges encountered when navigating in unstructured and dynamic environments. A CNN called DroNet, which is used to navigate quadrotors through the streets of a city, was proposed. Collecting data in urban areas within a city to train UAVs is dangerous for both pedestrians and vehicles, even if an expert pilot flies the quadrotor. Therefore, publicly available datasets from Udacity's project were used to learn the steering angles. The dataset includes over 70,000 driving car images collected and classified through six experiments. Five experiments are for training and one is for testing. A collision probability dataset was also developed for different areas of a city by placing a GoPro camera on the handlebars of a bicycle. UAV control is achieved via commands issued based on the output of DroNet. The collision probability is used to determine the quadrotor forward velocity. The desired yaw angle in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$ is determined by using predicted scaled steering in the range $[-1, 1]$. DroNet worked successfully to avoid unexpected situations and obstacles, predicting the collision probability and the desired steering angle. The quadrotor learned to fly in several environments, including in indoor environments, such as parking lots and corridors.

Smolyanskiy et al. [26] focused on autonomously navigating a micro aerial vehicle (MAV) in unstructured outdoor environments. A DNN called TrailNet was developed and used to estimate the view orientation and lateral offset of the MAV with respect to the center of a trail. A DNN-based controller allows for a stable flight and avoids overconfident maneuvers by utilizing a loss function that includes both label smoothing and an entropy reward. The MAV includes two vision modules: a second DNN and a visual odometry component that is called direct sparse odometry (DSO). The second DNN helps detect objects in the environment; the DSO estimates the depth to compute a pseudo-colored depth map. Their combination with TrailNet provides an autonomous flight controller that functions in unstructured environments. ResNet-18, SqueezeNet (a miniature version of AlexNet), the DNN architecture in [20], and TrailNet were compared considering autonomous long-distance flight ability, prediction accuracy, computational efficiency, and power efficiency. Only TrailNet is 100% autonomous; SqueezeNet and mini AlexNet are the closest to TrailNet at 98% and 97%, respectively. The least autonomous architecture is the DNN in [20] at 80%. The software modules run simultaneously in real time, and the quadcopter successfully flies in unstructured environments.

Cardenas et al. [30] developed a DNN-based flight position controller using a supervised DL technique. A dataset that includes position, velocity, acceleration, and motor output signals for different trajectories was created by using a PID flight controller. Five different NN architectures (from the literature) were utilized to learn the rotor speeds using the dataset, and their performance was compared. The five developed architectures are (i) ANN, (ii) ANN Feedback, (iii) LSTM, (iv) LSTM Layers interleaved with convolutional 1D layers (LSTMCNN), and (v) Convolutional 1D Layers cascaded with LSTM layers (CLSTM). A comparative study showed that LSTMCNN gives the best performance as a

DNN-based flight position controller. The LSTMCNN performance was checked against the PID position controller, and it was shown that LSTMCNN has a wider operational range than the PID position controller.

In summary, this part of the survey paper highlights applications of deep learning algorithms in multirotor UAVs. The focus is on navigation and control tasks such as hovering, ground target tracking, and trajectory tracking. Different DL techniques are explored, including DNNs, CNNs, and LSTM models, for altitude control, user tracking, trajectory tracking, and navigation in structured and unstructured environments. The reviewed research also compares DL-based controllers with traditional methods like PID controllers, showcasing improved performance and autonomy in UAV operations.

3.3. Reinforcement Learning

Offline RL has been extensively applied to multirotor UAVs. The literature review reveals 42 papers that focus on 13 different tasks, which include trajectory tracking, landing, navigation, formation control, flight control, and hovering.

In general, the RL framework uses an agent that is trained through trial and error to decide on an action that maximizes a long-term benefit. RL is described by a Markov decision process (MDP). The agent–environment interaction in an MDP is illustrated in Figure 5, where the agent, environment, and action represent, in engineering terms, the controller, the controlled system, and the control signal, respectively [12].

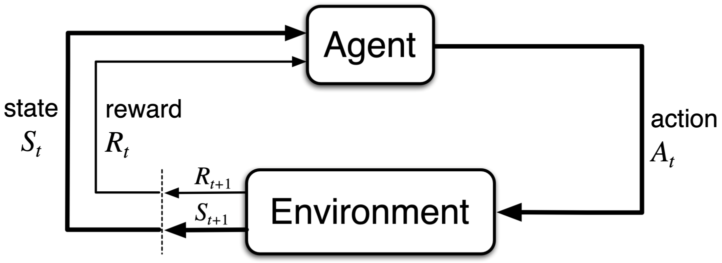

Figure 5. Block diagram and interaction between agent and environment [12].

RL algorithms are classified according to whether they are model-based or model-free, on-policy or off-policy, value-function-based or policy-search-based, or whether they are derived for planning or learning purposes. In what follows, the classification is in terms of value-function-based, policy-search-based, or actor–critic. Table 2 offers a summary of the different variations of RL algorithms.

Table 2. Classification of reinforcement learning (offline).

Methods	Algorithms	Papers
Value-function-based	Q-learning	Guerra et al. [50], Pham et al. [39], Kulkarni et al. [52], Abo et al. [59], Zeng and Xu [46]
	DQN	Xu et al. [16], Polvara et al. [32], Castro et al. [66], Shurrab et al. [68], Wu et al. [44], Kersandt et al. [38]
	LSPI	Vankadari et al. [37], Lee et al. [36], Srivastava et al. [43]
	IRL	Choi et al. [33]
	Others	Imanberdiyev et al. [31], Ye et al. [61], Farsi and Liu [63], Li et al. [57], Xia et al. [64]
Policy-search-based	PPO	Kooi and Babuška [54], Bhan et al [56]
	TRPO	Manukyan et al. [42]
	PILCO	Yoo et al. [18]
	PLATO	Kahn et al. [34]
	Others	Hu and Wang [53], Lambert et al. [41]
Actor–critic	DDPG	Jiang and Song [58], Rodriguez et al. [17], Rubi et al. [47], Rubi et al. [55], Ma et al. [65], Mitakidis et al. [67]
	TD3	Jiang and Song [58], Kooi and Babuška [54], Li et al. [51], Panetsos et al. [60]
	SAC	Jiang and Song [58], Kooi and Babuška [54],
	Fast-RDPG	Wang et al. [45]
	DeFRA	Li et al. [75]
	CdRL	Wang and Ye [62]
	Others	Pi et al. [48], Hwangbo et al. [35]

3.3.1. Value-Function-Based Algorithms

Value-function-based methods use state-value and action–state functions that are presented in (1) and (2), respectively, shown next:

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] for\ all\ s \in S \tag{1}$$

$$q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right] \tag{2}$$

where $v_\pi(s)$ denotes the *value function* for policy π at state s, while $q_\pi(s,a)$ represents the *action-value function* for policy π at state s and action a. $\mathbb{E}_\pi[\cdot]$ denotes the expected value under policy π. $\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ is the sum of discounted future rewards starting from time t in state s and represents the expected *discounted return*, and γ is the *discount rate*, $0 \leq \gamma \leq 1$, here. S_t and A_t represent the state and action at time t, respectively [12,76,77].

The discount rate γ plays a critical role in calculating the present value of future rewards. If $\gamma < 1$, the infinite sum has a finite value when the reward sequence, R_k, is bounded. If $\gamma = 0$, the agent cannot calculate future rewards; it can only calculate the immediate reward ($\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = 0^0 R_{t+1} + 0^1 R_{t+2} + \cdots = R_{t+1}$). So, the agent learns how to choose the action A_t to maximize R_{t+1}. If γ approaches 1, the future rewards are highlighted in the expected discount return; that is, the agent behaves in a more farsighted manner. For example, in [12] the chosen discount factor is close to 1. Li et al. [57] and Hu and Wang [53] chose a value of 0.9, and Castro et al. [66] and Panetsos et al. [60] chose a discount factor value of 0.99.

Value-function-based algorithms consist of variance algorithms, such as dynamic programming (DP), Monte Carlo (MC), and Temporal Difference (TD). The two most popular DP methods, policy iteration and value iteration, benefit from policy evaluation and policy improvement. The MC method is also based on policy evaluation and policy improvement, but unlike the DP method, an alternative policy evaluation process is utilized. The policy evaluation in the DP method employs a bootstrapping technique, while sampling and average return techniques are applied in the MC method. The TD method is created by combining the DP and MC methods applying sampling and bootstrapping [12,76,78].

TD has been extensively applied in control algorithms for multirotors executing diverse tasks, like landing, navigation, obstacle avoidance, path planning, and trajectory optimization. Q-learning and Deep Q-Networks (DQNs) stand out as commonly employed RL algorithms within the TD framework, particularly in value-function-based algorithm; see Guerra et al. [50], Pham et al. [39], and Polvara et al. [32].

Xu et al. [16] applied an end-to-end control scheme that includes a DNN and a double DQN algorithm for quadrotor landing on a stable platform. The output of the underlying deep reinforcement learning (DRL) model is the quadrotor speed in x and y, while the velocity in the z-direction is not controlled: it is considered fixed; this makes the problem easier. After testing, the improved DQN method produces good results on autonomous landing.

Imanberdiyev et al. [31] used a model-based RL algorithm on a quadrotor to create an efficient path to reach a destination by considering its battery life. The agent uses three states: the position in x and y and the battery level. The agent follows one of eight possible actions, moving on the x-y plane and learning the moving direction. The direction action is converted to trajectory commands that are executed using position control. In model-based RL methods, there is a limited number of actions the agent learns to create a sufficiently accurate environment model, as opposed to model-free RL methods. Model-based RL algorithms are not suitable for real-time systems since the planning and model learning aspects are computationally expensive. However, in [31], a parallel architecture called TEXPLORE is used. Thus, it is possible to take action fast enough based on the current policy: there is no need to wait for the planning and model update. Simulation results illustrate that the approach has the ability to learn and to perform well after a few iterations and to also perform actions in real time. Its performance was compared with that of Q-learning algorithms. In 150 episodes, while there was no significant change in the average reward in Q-learning, the average reward of TEXPLORE dramatically increased after the 25th episode. TEXPLORE obtained significantly more rewards than Q-learning in each episode.

In [46], the path design problem for a cellular-connected UAV was handled to reduce the mission completion time. A new RL-based UAV path planning algorithm was derived, and TD was applied to directly learn the state-value function. A linear function was added to the algorithm with tile coding. Function approximation has two advantages over table-based RL. It learns the parameter vector, which has a lower dimension than the state vector, instead of storing and updating the value function for all states. It also allows for generalization. Tile coding is used to build the feature vector. The parameter vector may be updated to minimize its mean squared error based on a stochastic semi-gradient method with a linear approximation for each state–reward–next-state transition observed by the agent. It is shown that TD with a tile coding algorithm overcomes problems with cellular networks in complex urban environments of size 2 km × 2 km with high-rise buildings. Also, the accumulated rewards from TD and TD with tile coding learning algorithms are almost identical, but tile coding provides faster convergence. When tested, the UAV reached the desired location without running into the coverage holes of cellular networks.

The approach discussed in [32] uses two DQNs. One is utilized for landmark detection, and the other is used to control the UAV's vertical descent. A hierarchy representing sub-policies is applied to the DQNs to reach decisions during the different navigation phases. The DQNs can autonomously decide on the next state. However, the hierarchy decreases the sophistication of the task decision. The algorithm performance was compared with an augmented reality (AR) tracker algorithm and with human pilots. The proposed algorithm is faster than a human pilot when landing on a marked pad but also more robust than the AR tracker in finding the marker. The complete details of [32] can be found in [79].

Ye et al. [61] developed a DRL-based control algorithm to navigate a UAV swarm around an unexplored environment under partial observations. This can be accomplished by using GAT-based FANET (GAT-FANET), which is a combination of the flying ad hoc network (FANET) and the graph attention network (GAT). Partial observations lead to a loss of information. Thus, a network architecture named Deep Recurrent Graph Network (DRGN) was developed and combined with GAT-FANET to collect environment spatial information and use previous information from memory via a gated recurrent unit (GRU). A maximum-entropy RL algorithm, called the soft deep recurrent graph network (SDRGN), was developed, which is a multi-agent deep RL algorithm. It learns a DRGN-based stochastic policy with a soft Bellman function. The performance of the DRGN (a deterministic model) and the SDRGN were compared with that of DQN, multi-actor attention critic (MAAC), CommNet, and graph convolutional RL (DGN). In a partially observable environment, the stochastic policy approach is more robust than the deterministic policy one. Also, GAT-FENAT provides an advantage because of its memory unit. When the number of UAVs increases, more information is required from the GAT-FANET, and this reduces the dependency on the memory unit. Results [61] show that policies based on GAT-FANET provide better performance in coverage than other policies. It is observed that graph-based communication improves performance in cooperative exploration and path planning, too. The SDRGN algorithm has lower energy consumption than DRGN, but DQN has the lowest energy consumption when compared with DRL methods. SDRGN and DRGN performance increases linearly with the number of UAVs. SDRGN shows better performance than DRL methods: this verifies that it has better transferability. Consequently, overall, SDRGN has better performance, scalability, transferability, robustness, and interpretability than other DRL methods.

Abo et al. [59] solved the problem of UAV landing on a dynamic platform by taking advantage of Q-learning. Two types of adaptive multi-level quantization (AMLQ) were used: AMLQ 4A with four actions and AMLQ 5A with five actions; they were then compared with a PID controller. The PID position magnitude errors in x and y were higher than the corresponding AMLQ errors, while the oscillation in the AMLQ models was higher than in the PID controller. The developed AMLQ reduces the error on the targeted landing platform. This solution provides faster training and allows for knowledge representation without the need for a DNN.

Path planning is effectively used in several areas that include precision agriculture. Castro et al. [66] worked on adaptive path planning using DRL to inspect insect traps on olive trees. The proposed path planning algorithm includes two parts: the rapidly exploring random tree (RRT) algorithm and a DQN algorithm. The former searches for path options, while the latter performs optimized route planning, integrating environmental changes in real time; however, the training process of DQN is completed offline. Simulation runs were performed in an area of 300 m^2 with 10 dynamic objects; the UAV was provided with a safe route, determined by the proposed approach, and it arrived at the insect traps to take their picture.

Shurrab et al. [68] studied the target localization problem and proposed a Q-learning-based data-driven method in which a DQN algorithm helps overcome dimensionality challenges. Data measurements from the previous and current steps, the previous action, and the direction of the nearest boundary of the UAV compose the state space. The action space includes the UAV linear velocity and the yaw angle that determines the flight direction. This approach was compared with the traditional uniform search method and the gradient descent-based ML technique; it returned better results in terms of localization and traveled distance.

Guera et al. [50] emphasized detection and mapping for trajectory optimization. For detection, the aim is to minimize 'wrong detection', and for mapping, the aim is to minimize the uncertainty related to estimating the unknown environment map. The proposed MDP-based RL algorithm, inspired by Q-learning, consists of state and control estimations. The states are the UAV position depending on actions, a binary parameter that shows the presence or absence of a signal source in the environment, and the states of each cell. The action space includes the control signal to move the UAV from one cell to another in the grid (environment) map. Numerical results show that this technique provides a high probability of target detection and improves the capabilities of map exploration.

Pham et al. [39] handled the UAV navigation problem using Q-learning. The navigation problem was formulated using a discretized state space within a bounded environment. The algorithm learns the action, which is the UAV moving direction in the described environment. The state space includes the distance between the UAV and the target position, and the distance to the nearest obstacle in the north, south, west, or east direction. UAV navigation following the shortest path was demonstrated.

Kulkarni et al. [52] also used Q-learning for navigation purposes. The objective is to determine the location of a victim by using an RF signal emitted from a smart device. The transmitted signal reaches the agent, and according to the received signal strength (RSS), the agent learns to choose one of eight directions separated by 45 degrees on the x-y plane. For mapping, a grid system is utilized, and each state label is correlated to a particular RSS value (two adjacent grids in the map have different RSS values). Each location on the map has a unique state. The ϵ-greedy approach provides an action to the UAV, and each episode or iteration is completed when the RSS value of the grid is determined to be greater than -21 dBm: this value means that the distance from the victim is less than 2 m. The proposed approach was tested for different starting positions on different floor plans, demonstrating that the UAV successfully reached the victim's position.

Choi et al. [33] trained a multirotor UAV by mimicking the control performance of an expert pilot. A pilot collects data from several actual flights. Then, a hidden Markov model (HMM) and dynamic time warping (DTW) are utilized to create the trajectory. Inverse RL is used to learn the hidden reward function and use it to design a controller for trajectory following. Simulations and experiments showed successful results.

Wu et al. [44] worked on the general task of 'object finding', for example, in rescue missions. A DQN algorithm is used for trajectory planning. The elimination of the loop storm effect that reflects the current sequence in an MDP is repeated, and it does not cause a punishment in continued actions. The Odor Storm that is caused by not reaching the highest reward value when the agent gets closer to the target increases the convergence speed of the training process. It is shown that the breakout loop storm technique and the odor effect reduce the training process time.

In Kersandt et al. [38], a DNN is trained with a DRL algorithm to control a fully autonomous quadcopter that is equipped with a stereo-vision camera to avoid obstacles. Three different DRL algorithms, DQN, double DQN (DDQN), and Dueling DDQN, are applied to the system. The average performance of each algorithm with respect to rewards is 33, 120, and 116, respectively; they are all below human performance. The results of applying DDQN and Dueling techniques show that the quadrotor reaches the target with 80% success.

Liu et al. [40] and Zhao et al. [49] used RL for formation control. In [40], a value-function-based RL control algorithm was applied to leader–follower quadrotors to tackle the attitude synchronization problem. The output of each quadrotor is synchronized with the output of the leader quadrotor by the designed control system. In [49], the aim was to solve the model-free robust optimal formation control problem by utilizing off-policy value-function-based algorithms. The algorithms are trained for robust optimal position and attitude controllers by using the input and output data of the quadrotors. The theoretical analysis and simulation results matched, and the robust formation control method worked effectively.

Vankadari et al. [37] and Lee et al. [36] worked on the landing task using a Least-Square Policy Iteration (LSPI) algorithm that is considered a form of approximate dynamic programming (ADP). Srivastava et al. [43] and Li et al. [57] applied an LSPI algorithm in multirotor UAVs for target tracking and trajectory planning, respectively. ADP is used to solve problems with large state or action spaces. ADP approximates the value function or policy with function approximation techniques, since storing values for every state–action pair is not practical.

In [37], an LSPI algorithm was used to study the landing problem. An RL algorithm estimates quadrotor control velocities using instantaneous position and velocity errors. The optimal value function of any policy is determined by solving the Bellman equation, as applied to a linear system. In the RL algorithm, the LSPI method forecasts the value function, parameterizing it into basis functions instead of calculating an optimal value function. The RL algorithm converges quickly and learns how to minimize the tracking error for a given set point. Different waypoints are used to train the algorithm for landing. The method can also be used effectively in noisy environments. Simulations and real environment results demonstrate the applicability of the approach.

The research in [63] provided a low-level control approach for a quadrotor by implementing a structured online learning-based algorithm (SOL) [80] to fly and keep the hovering position at a desired altitude. The learning procedure consists of two stages: The quadrotor is first flown with almost equal pulse-width modulation (PWM) values for each rotor; these values are collected to create an initial model. Then, learning in a closed-loop form is applied using the initial model. Before applying closed-loop learning, three pre-run flights are completed, with 634 samples collected in 68 s of flying. The state samples are determined at each time step in the control loop, and then the system model is updated using an RLS algorithm. After determining the updated model, a value function (needed to find the control value for the next step) is updated. The quadrotor is autonomously controlled. This online learning control approach successfully reaches the desired position and keeps the quadrotor hovering.

In [36], a trained NN was adopted for guidance in a simulation environment. A quadrotor with a PID controller and an onboard ground-looking camera was used. The camera provides the pixel deviation of the targeted landing platform from an image frame, and a laser rangefinder procures the altitude information. During training, the NN is trained to learn how to control the UAV attitude. In simulation studies, the UAV reached the proposed landing location. In experiments, the AI pilot was turned off below an altitude of 1.5 m, but the AI pilot could land at the targeted location using a vision sensor. The trajectories were not smooth because the landing location in the image was not accurately determined due to oscillations, because image processing errors occurred in the actor NN, because signal transmission created a total delay of 200 ms, and because there are disturbances in real-world environments.

Three target tracking approaches that deserve attention are the Image-based (IBVS), Position-based (PBVS), and Direct Visual Servoing approaches. In Kanellakis and Nikolakopoulos [81], IBVS was found to be the more effective approach for target tracking since it directly tackles the control problem in the image space; it also has better robustness when it comes to camera calibration and to depth estimation errors. Srivastava et al. [43] tracked a maneuvering target using only vision-based feedback, IBVS. However, tracking is difficult when using only monocular vision without depth measurements. This deficiency is eliminated by an RL technique where optimal control policies are learned by LSPI to track the target quadrotor. Two different basis functions (with and without velocity basis) and four types of reward functions (only exponential reward, quadratic reward function without velocity control, quadratic reward function with velocity control, asymmetric reward function) are described in [43]. The basis function with a velocity basis shows better performance than the basis function without a velocity basis.

In [57], the objective is to solve the problem of cable-suspended load transportation utilizing three quadrotors. The trajectory planning method is based on a value-function approximation algorithm with the aim to reach the final position as fast as possible while keeping the load stable. This method includes two processes: trajectory planning and tracking. The trajectory planning process consists of parameter learning and trajectory generation. Training and learning help determine the parameter vector of the approximate value function (parameter learning part). In the trajectory generation phase, the value function is approximated by using the learned parameters from the former stage, and the flight trajectory is determined via a greedy strategy. The effectiveness of the load trajectory and the physical effect on the quadrotor flight were checked based on the trajectory tracking process. The quadrotors are independent; in the trajectory tracking phase, positions and attitudes are controlled with a hierarchical control scheme using PID controllers (transmitting the position of the load to the controller of the quadrotor). The results show that the actual value function is successfully estimated. Also, the value function confirms that the proposed algorithm works effectively.

Xia et al. [64] use an RL control method for autonomous landing on a dynamic target. Unlike other studies, position and orientation constraints for safe and accurate landing are described. Adaptive learning and a cascaded dynamic estimator are utilized to create a robust RL control algorithm. In the adaptive learning part, the critic network weight is formulated and calculated in an adaptive way. Also, the stability of the closed-loop system is analyzed.

Therefore, this part of the survey demonstrates how value-function-based RL algorithms are implemented and tested for UAV navigation and control. In detail, Xu et al. [16] integrated a DNN with a double DQN algorithm for quadrotor landing, yielding improved results. Imanberdiyev et al. [31] employed model-based RL for efficient path planning, outperforming traditional methods. Zeng et al. [46] introduced an RL-based algorithm with tile coding for UAV path planning, showcasing enhanced convergence. Polvara et al. [32] proposed a hierarchical RL approach using dual DQNs for UAV landing, exhibiting robust performance. Ye et al. [61] developed a DRL-based control algorithm for UAV swarm navigation, highlighting the efficacy of graph-based communication. Abo et al. [59] studied UAV landing on dynamic platforms using Q-learning with adaptive quantization, achieving improved accuracy. Other reviewed studies explored RL algorithms for path planning, trajectory optimization, target tracking, and formation control, with the aim to illustrate the versatility and efficacy of RL techniques. Overall, value-function-based RL methods offer powerful tools for UAV navigation and control, enabling efficient decision-making and adaptation to dynamic environments. These methods continue to be refined for a wide range of UAV tasks, promising advancements in the underlying UAV technology.

3.3.2. Policy-Search-Based Algorithms

Value-function-based methods calculate the value of an agent's every possible action to choose the one based on the best value. The probability distribution over all available actions plays a key role in policy-based methods and in the agent's decision about the action at each time step. A comparison of value-function-based and policy-search-based algorithms is provided in Table 3.

Table 3. Comparison of value-function-based and policy-search-based methods.

Value-Function-Based	Policy-Search-Based
Indirect policy optimization	Direct policy optimization
Generally off-policy	On-policy
Simpler algorithm	Complex algorithm
Computationally expensive	Computationally inexpensive
More iterations to converge	Fewer iterations to converge

Kooi and Babuška [54] developed an approach using deep RL to land a quadrotor on an inclined surface autonomously. Proximal Policy Optimization (PPO), Twin-Delay Deep Deterministic Gradient (TD3), and Soft Action–Critic (SAC) algorithms were applied to solve this problem. The TD3 and SAC algorithms successfully trained the set-point tracking policy network, but the PPO algorithm was trained in a shorter time and provided a better performance on the final policy. Trained policies may be implemented in real time.

Hu and Wang [53] utilized an advanced PPO RL algorithm to find the optimal stochastic control strategy for a quadrotor speed. During training, actor and critic NNs are used. They have the same nine-dimensional state vector (Euler angles, Euler angle derivatives, errors between expected and current velocities after integration on the x, y, and z axes). An integral compensator is applied to both NNs to improve speed-tracking accuracy and robustness. The learning approach includes online and offline components. In the offline learning phase, a flight control strategy is learned using a simplified quadrotor model, which is continuously optimized in the online learning phase. In offline learning, the critic NN evaluates the current action to determine an advantage value choosing a higher learning rate to improve evaluation. In online learning, the action NN is composed of four policy-trainable sub-networks. The state vector is used as input to the four sub-networks; their outputs are the mean and variance of the corresponding four Gaussian distributions, each normalized to [0, 1]. Parameters of the four policy sub-networks are also used in the old policy networks that are untrainable. The old policy sub-network parameters are fixed. The four policy sub-networks in the action NN are trained to produce new actions in the next batch. When applying new actions to the quadrotor, new states are recorded in a buffer. After the integration and compensation process, a batch of the state vector is used as input to the critic NN. The batch of the advantage values is the output of the critic NN; it is used to evaluate the quality of the actions taken to determine these states. The parameters of the critic NN are updated by minimizing the advantage value per batch. The policy network is updated per batch using the action vectors taken from the old policy network, the state vector from the buffer, and the advantage value from the critic NN.

In [53], the PPO and the PPO-IC algorithms were compared with the offline PPO one and a well-tuned PID controller. The average linear velocity steady-state error of the PPO-IC approaches zero faster, and it is smaller than that of PPO. The average accumulated reward of the PPO-IC reaches a higher value. The PPO-IC converges closer to the targeted velocity on the x-, y-, and z-axes than PPO. PPO-IC velocity errors on the x-, y-, and z-axes are much smaller compared to the PPO errors. The Euler angle errors are also smaller in the PPO-IC algorithm. In the offline learning phase, the nominal quadrotor weight is increased by 10% in each step until it reaches 150% of the nominal weight. The performance of the well-tuned PID controller and the proposed method were compared.

When the quadrotor weight increased, the velocity error along the z-axis increased, too, but the PPO-IC algorithm demonstrated stable behavior without fluctuations in speed tracking. Moreover, 12 experiments were conducted when the nominal 0.2 m radius of the quadrotor was increased from 50% to 550%, that is, from 0.1 m to 1.1 m. PID and PPO-IC performed similarly when the radius was between 0.2 and 0.4 m. For higher values, the PID performance decreased, even when convergence to the desired value was observed. However, the PID controller could not control the quadrotor: it became unstable when the radius increased to more than 1 m. On the contrary, changes in the radius value slightly affected the performance of the PPO-IC algorithm.

Kahn et al. [34] and Bhan et al. [56] worked on failure avoidance and on compensating for failures occurring during flights. In [34], a Policy Learning using Adaptive Trajectory Optimization (PLATO) algorithm, a continuous, reset-free RL algorithm, was developed. In PLATO, complex control policies are trained with supervised learning using model predictive control (MPC) to observe the environment. Partially trained and unsafe policies are not utilized in the action decision. During training, taking advantage of the MPC robustness, catastrophic failures are minimized since it is not necessary to run the learned NN policy during training time. It was shown that good long-horizon performance of the resulting policy was achieved by the adaptive MPC. In [56], accommodation and recovery from fault problems occurring in an octacopter were achieved using a combination of parameter estimation, RL, and model-based control. Fault-related parameters are estimated using an Unscented Kalman Filter (UKF) or a Particle Filter (PF). These fault-related parameters are given as inputs to a DRL, and the action NN in the DRL provides a new set of control parameters. In this way, the PID controller is updated when the control performance is affected by the parameter(s) correlated with faults.

In [42], a DRL technique was applied to a hexacopter to learn stable hovering in a state–action environment. The DRL used for training is a model-free, on-policy, actor–critic-based algorithm called Trust Region Policy Optimization (TRPO). Two NNs are used as nonlinear function approximators. Experiments showed that such a learning approach achieved successful results and facilitated controller design.

Yoo et al. [18] combined RL and deterministic controllers to control a quadrotor. Five different methods, the original probabilistic inference for learning control (PILCO), PD-RL with high gain, PD-RL with low gain, LQR-RL, and LQR-RL with model uncertainty, were compared via simulations for when the quadrotor tracks a circular reference trajectory. The high-gain PD-RL approaches the reference trajectory quickly. The low-gain PD-RL behaves less aggressively and reference trajectory tracking is delayed. The convergence rates of the PD-RL and LQR-RL methods are better. The performance is also better when compared to the original PILCO. The main advantages of combining a deterministic controller with PILCO are simplicity and rapid learning convergence.

In [41], errors on the pith and roll angles were minimized to provide stability during hovering. A user-designed objective function uses simulated trajectories to choose the best action. The objective function also minimizes the cost of each state. The performance of this controller is worse than a typical quadrotor controller's performance. However, the proposed controller achieved hovering for up to 6 s after training using 3 min of data.

Thus, this part of the survey focuses on value-function-based and policy-based methods. In detail, Kooi and Babuška [54] employed deep RL algorithms, PPO, to autonomously land quadrotors on inclined surfaces, demonstrating PPO's superior performance. Hu and Wang [53] utilized an advanced PPO algorithm to optimize stochastic control strategies for quadrotor speed, outperforming traditional PID controllers. Kahn et al. [34] and Bhan et al. [56] addressed failure avoidance and fault compensation in UAV flights using RL and model-based control techniques. Other techniques integrate RL with deterministic controllers, enhancing trajectory tracking and stability during flight maneuvers. All reviewed methods collectively showcase the effectiveness of policy-search-based RL techniques in overcoming challenges in UAV control, from autonomous landing to fault tolerance and trajectory tracking.

3.3.3. Actor–Critic Algorithms

Actor–critic algorithms consist of both value-function-based and policy-search-based methods. The actor refers to the policy-search-based method and chooses the actions in the environment; the critic refers to the value-function-based method and evaluates the actor using the value function.

In [58], three different RL algorithms, DDPG, TD3, and SAC, were applied to study multirotor landing. Using the DDPG method did not result in successful landings. The TD3 and SAC methods successfully completed the landing task. However, TD3 required a longer training period and landing was not as smooth, most likely because of noise present in the algorithm.

Rodriguez et al. [17] studied landing on a dynamic/moving platform using DDPG. Slow and fast scenarios were tried in 150 test episodes. During the slow scenario, the moving platform (the moving platform trajectory was periodic) velocity was 0.4 m/s, and during the fast scenario, it was set to 1.2 m/s. The success rates were 90% and 78%, respectively. Using a constant velocity on the z-axis resulted in landing failure on the moving platform. This problem may be overcome by using the velocity on the z-axis as a state, but this makes the training process more complicated, and learning the landing process becomes more challenging.

Rubi et al. [47] solved the quadrotor path-following problem using a deep deterministic policy gradient (DDPG) reinforcement learning algorithm. A lemniscate and one lap of a spiral path were used to compare agents with different state configurations in DDPG and in an adaptive Nonlinear Guidance Law (NLGL) algorithm. The agent has only two states: distance error and angle error. According to the results, the adaptive NLGL has a lower distance error than the two-state agent, but its distance error is significantly greater than that of the agent with the future states on the lemniscate path.

Rubi et al. [55] also used three different approaches to solve the path-following problem using DDPG. The first agent utilizes only instantaneous information, the second uses a structure (the agent expects the curve), and the third agent computes the optimal speed according to the shape of the path. The lemniscate and spiral paths were used to test the three agents. The lemniscate path was used in the training and test phases. The agents were evaluated in tests but with the assumption that the third agent is also limited by a maximum velocity of 1 m/s. For the lemniscate path, the agents were first tested with ground-truth measurements. The second agent showed the best performance with respect to cross-track error. When the agents were tested with the sensor model, the third agent showed slightly better performance in terms of cross-track errors. Then, all agents were tested in the spiral path. When the performance of the agents was compared in simulations with ground-truth measurements and with sensor models, the third agent (with a maximum velocity of 1 m/s) showed the best performance in terms of position error. In all tests, the third agent (without a maximum velocity limitation) completed the tracks faster.

Wang et al. [45] handled the UAV navigation problem in a large-scale environment using DRL. Two policy gradient theorems within the actor–critic framework are derived to solve the problem, which is formulated as a partially observable Markov decision process (POMDP). As opposed to conventional navigation methods, raw sensor measurements are utilized in DRL; control signals are the output of the navigation algorithm. Stochastic and deterministic policy gradients for POMDP are applied to the RL algorithm. The stochastic policy requires samples from both the state and action spaces. The deterministic policy requires only samples from the state space. Therefore, the RL algorithm with a deterministic policy is faster (and preferred): it is called a fast recurrent deterministic policy gradient algorithm (Fast-RDPG). For comparisons, four different large-scale complex environments were built with random-height buildings to test the DDPG, RDPG, and Fast-RDPG. The success rate of the Fast-RDPG was significantly higher in all environments. Fast-RDPG had the lowest crash rate in one environment. DDPG provided the best performance with respect to the average crash rate in all environments. Fast-RDPG had a much lower crash rate than RDPG. However, Fast-RDPG provided a much lower stray rate than the other algorithms in all environments.

Li et al. [75] developed a new DRL-based flight resource allocation framework (DeFRA) for a typical UAV-assisted wireless sensor network used for smart farming (crop growth condition). DeFRA reduces the overall data packet loss in a continuous action space. A DDPG is used in DeFRA, and DeFRA learns to determine the instantaneous heading and speed of the quadrotor and to choose the ground device to collect data from the field. The time-varying airborne channels and energy arrivals at ground devices cause variations in the network dynamics. The network dynamics are estimated by a newly developed state characterization layer based on LSTM in DeFRA. An MDP simultaneously handles the control of the quadrotor's maneuver and the communication schedule according to decision parameters (time-varying energy harvesting, packet arrival, and channel fading). The state space comprises the battery level, the data buffer length of all ground devices, the battery level and location of the UAV, the channel gain between the UAV and the ground devices, and the time-span parameter of the ground device. The UAV's current battery level depends on the battery level of the UAV in the previous time step, harvested energy, and energy consumption. The quadrotor is required to keep its battery level equal to or higher than the battery level threshold. The performance was compared with two DRL-based policies, DDPG-based Movement Control (DDPG-MC) and DQNs-based Flight Resource Allocation Policy (DQN-FRA), and with two non-learning heuristics, Channel-Aware Waypoint Selection (CAWS) and Planned Trajectory Random Scheduling (PTRS). DeFRA provides lower packet loss than other methods. The relation between the packet loss rate and the number of ground devices was investigated according to all methods. The DRL-based methods outperformed CAWS and PTRS. For up to 150 ground devices, DeFRA and DDPG-MC showed similar performance and were better than the other methods, but after increasing the number of ground devices to 300, DeFRA provided better performance than DDPG-MC.

Pi et al. [48] created a low-level quadrotor control algorithm to hover at a fixed point and to track a circular trajectory using a model-free RL algorithm. A combination of on-policy and off-policy methods is used to train an agent. The standard policy gradient method determines the update direction within the parameter space, while the TRPO and PPO algorithms are designed to identify an appropriate update size. However, for updating the policy, the proposed model establishes new updating criteria that extend beyond the parameter space concentrating on local improvement. The NN output provides the thrust of each rotor. The simulator was created in Python using the dynamic model of Stevens et al. [82]. The effects of the rotation matrix and quaternion were investigated in the learning process. The model with the quaternion may converge slower in the training process than the model with the rotation matrix. However, both models showed similar performance when tested.

Ma et al. [65] developed a DRL-based algorithm for trajectory tracking under wind disturbance. The agent learns to determine the rotation speed of each rotor of a hexacopter. A DDPG algorithm is used, but in addition to the existing DDPG algorithm, a policy relief (PR) method based on an epsilon-greedy exploration-based technique and a significance weighting (SW) method are integrated into the DDPG framework. The former method improves the agent's exploration skills and its adaptation to environmental changes. The latter helps the agent update its parameters in a dynamic environment. In training, the implementation of PR and SW methods in the DDPG algorithm provides better exploration performance and faster convergence of the learning process, respectively, even in a dynamic environment. This method reaches a higher average reward and has a lower position error compared to the DDPG, DDPG with RP, and DDPG with SW. Also, this algorithm provides higher control accuracy compared to the cascaded active disturbance rejection control algorithm in terms of position, velocity, acceleration, and attitude errors.

Hwangbo et al. [35] proposed a method to increase UAV stabilization. An NN improves UAV stability training with RL. Monte Carlo samples are produced by on-policy trajectories and are used for the value function. The value network is used to guide policy training, and the policy network controls the quadrotor. Both are updated in every iteration. A new analytical measurement method describes the distance between action distribution and a new policy for policy optimization. This policy network gives an accurate reaction to the step response. The policy stabilizes the quadrotor even under extreme situations. The algorithm shows better performance than DDPG and Trust Region Policy Optimization with a generalized advantage estimator (TRPO-gae) in terms of computation time.

Mitakidis et al. [67] also studied the target tracking problem. A CNN-based target detection algorithm is used on an octocopter platform to track a UGV. DDPG-RL is applied in a hierarchical controller (instead of a position controller). The CNN learns to detect the UGV, and the DDPG-RL algorithm learns to determine the roll, pitch, and yaw actions in the outer loop of the controller. These actions are taken from the NN output, normalized to the range $[-1, 1]$, but these normalized values are multiplied by a spectrum of acceptable values. While the roll and pitch actions span between -3 and 3 degrees, the yaw action ranges between -5 and 5 degrees. An experiment was conducted with a low-altitude octocopter and with the manual control of a UGV. Fluctuations were observed in the distance error due to the aggressive maneuver of the UGV, but overall, the results are good.

Li et al. [51] studied the target tracking problem in uncertain environments. The proposed approach consists of a TD3 algorithm and meta-learning. The used algorithm is named meta twin delay deep deterministic policy gradient (meta-TD3). TD3 learns to control the linear acceleration of the UAV and the angular velocity of the heading angle. The state space includes the position of the quadrotor in the x-y plane, the heading angle, the linear velocity, the angle between the motion direction and the straight line between the UAV and the target, and the Euclidean distance between the UAV and the target. Meta-learning overcomes the multi-task learning challenge. Tasks are trajectories of the ground vehicle that is followed. A reply buffer is built for the task experience. When the agent interacts with the environment, the state space, the action space, the reward value, and the next-step state space that corresponds to the task are saved into the reply buffer. The method provides a significant improvement in the convergence value and rate. Meta-TD3 adapts to the different movements of the ground vehicle faster than the TD3 and DDPG algorithms. Meta-TD3 tracks the target more effectively.

Panetsos et al. [60] offer a solution to the payload transportation challenge using a DRL approach. An attitude PID controller is used in the inner loop of the cascaded controller structure, while a position controller in the outer loop is replaced with a TD3-based DRL algorithm. The DRL algorithm in the outer loop learns to create the reference Euler angles, the roll and pitch, and the reference translational velocity of the octocopter on the z-axis. The method controls the system successfully to reach the desired waypoints.

Wang and Ye [62] developed consciousness-driven reinforcement learning (CdRL) for trajectory tracking control. The CdRL learning mechanism consists of online attention learning and consciousness-driven actor–critic learning. The former selects the best action. The latter increases the learning efficiency based on the cooperation of all subliminal actors. Two different attention-learning methods are utilized for online attention learning: short-term attention learning and long-term attention learning. The aim of the former is to select the best action. The latter selects the best action to sustain the system's stability. The long- and short-term attention arrays are combined to make a decision about which actor should be given more attention. This learning algorithm was compared with Q-learning; the position error in the proposed algorithm was lower than in Q-learning. The same was also seen in the velocity error. However, this method was slightly better than Q-learning when it comes to attitude error. The UAV was successfully controlled to track the desired trajectory by the CdRL algorithm.

Xu et al. [83] created a benchmark using PPO, SAC, DDPG, and DQN algorithms for single-agent tasks and multi-agent PPO (MAPPO), heterogeneous-agent PPO (HAPPO), multi-agent DDPG (MADDPG), and QMIX algorithms for multi-agent tasks with different drone systems. Single-agent tasks include hovering, trajectory tracking, and flythrough.

Multi-agent tasks cover hover, trajectory tracking, flythrough, and formation. To increase the task variation, the payload, inverse pendulum, and transportation challenges are integrated into the single- and multi-agent tasks. The learning performance differs based on specific tasks.

Thus, applications of actor–critic RL as applied to UAV control have been summarized in this subsection. Studies like [58] tested algorithms like DDPG, TD3, and SAC for multirotor landing, with SAC and TD3 performing well. Rubi et al. [47,55] examined quadrotor path following, while Wang et al. [45] focused on UAV navigation in complex environments, favoring Fast-RDPG. Li et al. [75] introduced DeFRA for UAV-assisted networks, outperforming heuristics. Pi et al. [48] addressed low-level quadrotor control, and Ma et al. [65] developed a DDPG-based algorithm for trajectory tracking under wind. Various other studies explored target tracking, payload transportation, and consciousness-driven RL for trajectory control, demonstrating RL's effectiveness in diverse UAV applications. Xu et al. [83] established a benchmark for UAV tasks, evaluating different RL algorithms' performance.

4. Online Learning

In online learning, an agent learns and is updated during the system operation by incorporating collected data from sensors to make and improve decisions while also interacting with the environment. The agent learns in real time, enhancing its decision and prediction capabilities (with respect to the assigned mission). Before proceeding, it is important to mention three more perspectives that further clarify what online learning is (compared to the definition provided in Section 2) and also offer a more 'rounded' and more 'open-minded' point of view to consider.

Srinivasan and Jain [84] stated that the intelligent behavior of an agent may be limited given the extreme difficulty in developing knowledge structures and rule bases, which completely describe the task of the agent if the task is complex by nature. This problem can be partially overcome by causing the agent to learn on its own during this task.

Hoi et al. [19] define online learning as a method of ML based on which data arrive in sequential order, and where a learner aims to learn and to update the best prediction for future data at every step. Online learning is able to overcome the drawbacks of batch learning because the predictive model can be updated instantly for any new data instances.

According to Otterlo and Wiering, online learning is defined as "an important aspect during the learning task (during the learning process) is the distinction between online and offline learning. The difference between these two types is influenced by factors such as whether one wants to control a real-world entity or whether all necessary information is available. Online learning performs learning directly on the problem instance. Offline learning uses a simulator of the environment as a cheap way to get many training examples for safe and fast learning" [85].

The literature review (from 2015) reveals that 11 papers have used online-learning-based algorithms to control multirotor UAVs in particular quadrotors. Their review and comparison provide a more accurate understanding of what the research is all about, what is being learned, why and how it is learned, and what results have been produced.

Table 4 summarizes online learning approaches that have been applied to multirotor UAVs. The table mainly includes the publication year of the paper, the adopted or derived learning model, the application task/domain, the advantages of the algorithm, and the approaches that the proposed algorithm is compared with.

Yang et al. [86] developed optimal control protocols to solve the distributed output synchronization problem for leader–follower multi-agent systems. The adopted RL algorithm solves the underlying non-homogeneous algebraic Riccati equations (AREs) in real time: this is basically a distributed optimal tracking problem. Solving the AREs guarantees the synchronization of the followers' and the leader's outputs. A distributed observer is derived to forecast the leader's state and to produce the reference signal for each follower. The advantage of the proposed RL algorithm is that it does not require knowledge of the quadrotor dynamics. This method gives better results than the adaptive control approach developed by Das and Lewis [87]. Moreover, when the transient response of the output synchronization error for each follower is compared, the error in [86] converges to zero faster than in [87].

A neural proactive closed-loop controller was developed in [88] that learns control parameters. The proposed solution provides two advantages. The first is computational inexpensiveness due to using only three neurons instead of a neural network. The second is that the learning process can be completed within a few trials. No knowledge of the UAV dynamic model is required. This technique was used for quadrotor speed adaptation and obstacle avoidance following the detailed block diagram configuration shown in Figure 6. For performance evaluation purposes, an MPC and the proposed method were implemented and compared. The MPC requires knowledge of the quadrotor dynamic model. Thus, the neural proactive controller method is more appealing. It also provides robust results for speed adaptation even in the presence of wind disturbances. This approach was implemented and tested on UAVs with different dynamics in a Gazebo environment with different maximum flying speeds. The neural proactive controller generates a control command 56.32% faster (on average) than the MPC; it is also 99.47% faster than the MPC in total learning and optimization time. The UAV is trained within 3–4 trials, adapting its speed, and learns to stay away from obstacles at a safe distance.

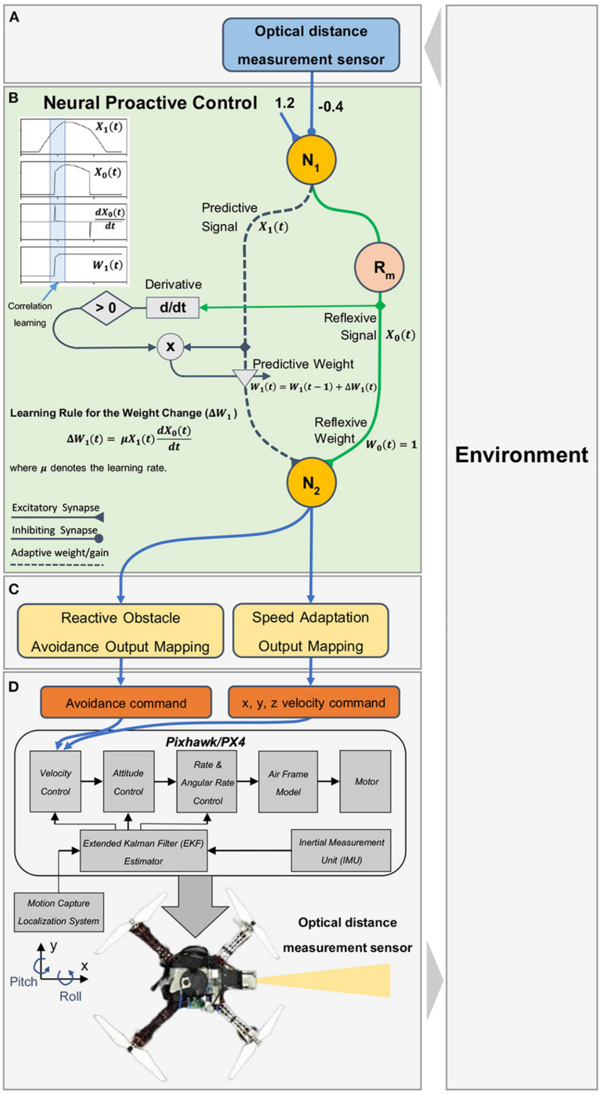
A
Optical distance measurement sensor
B
Neural Proactive Control
1.2
-0.4
$X_1(t)$
$X_0(t)$
$\frac{dX_0(t)}{dt}$
$W_1(t)$
N1
Predictive Signal
$X_1(t)$
Rm
Derivative
Correlation learning
> 0
d/dt
Reflexive Signal
$X_0(t)$
x
Predictive Weight
$W_1(t) = W_1(t-1) + \Delta W_1(t)$
Learning Rule for the Weight Change (ΔW_1)
$\Delta W_1(t) = \mu X_1(t) \frac{dX_0(t)}{dt}$
where μ denotes the learning rate.
Reflexive Weight
$W_0(t) = 1$
N2
Excitatory Synapse
Inhibiting Synapse
Adaptive weight/gain
C
Reactive Obstacle Avoidance Output Mapping
Speed Adaptation Output Mapping
D
Avoidance command
x, y, z velocity command
Pixhawk/PX4
Velocity Control
Attitude Control
Rate & Angular Rate Control
Air Frame Model
Motor
Extended Kalman Filter (EKF) Estimator
Inertial Measurement Unit (IMU)
Motion Capture Localization System
Optical distance measurement sensor
y
x
Pitch
Roll
Environment

Figure 6. Online learning controller scheme of [88]. (**A**) The input module. (**B**) The neural control module with the ICO learning mechanism. (**C**) The output mapping module. (**D**) The flight controller (Pixhawk/PX4).

Shin et al. [89] applied online learning to optimize speed parameters with the aim to complete missions in less time (i.e., drone races through gates). A pre-trained actor network, modified to be suitable for online training, detects a specific object. The control network, which comprises the online learning component, provides the UAV's optimized linear velocity and acceleration in the x-, y-, and z-directions, as well as the distance to a gate to pass through with maximum velocity and without collisions. Each gate and the gate center are detected by a monocular camera system. The UAV coordinates are determined based on the provided information, and a loss function is used to make sure that the quadrotor passes through the center of the gate. The maximum velocity, acceleration, and threshold distance (which determines whether or not the quadrotor reaches the gate location) are parameters that are optimized for each gate. They are updated each time the quadrotor passes through each gate. As opposed to several conventional navigation approaches, this method requires a single neural network to control the quadrotor. The evolution-based approach is combined with the gradient-based method, and this cooperation provides an advantage to increase the system performance in drone racing. The derived network successfully handled all race processes in environments with demanding obstacle avoidance and navigation requirements. When tested in real time, the quadrotor completed the race track in around 180 s on the first try, but in about 60 s after about 9000 times.

Sarabakha and Kayacan [90] proposed an online-learning-based control method to improve UAV trajectory tracking. The total thrust and the three torques around the x-, y-, and z-axes are used as control inputs. The online learning component learns to update the weight of a DNN to improve the control performance. This method consists of two phases: pre-training, called offline learning, and post-training, called online learning. In pre-training, supervised learning is used to control the quadrotor by mimicking a PID controller (using an input–output dataset for a set of trajectories). When the quadrotor is controlled by the trained DNN controller, a fuzzy logic system (FLS) keeps training the DNN online by providing feedback about its performance. The advantage of using the fuzzy mapping in the online learning phase is that the computation time is considerably reduced. Also, this provides an opportunity for real-time applicability. The offline-learning-based performance is not different from the classic PID controller performance. However, online learning allows for the system to accurately predict the evolution and desired signal estimation. This approach was tested in slow circular, fast circular, and square-shaped trajectories. The performance of the well-tuned PID controller used in offline training, the offline-trained DNN, and the online training approach was compared, with the last one clearly showing better performance for the slow and fast circular trajectories. The square-shaped trajectory was not used in the offline training phase. When the three were implemented and tested in the square-shaped trajectory, again, the online learning method slightly outperformed the other approaches.

O'Connell et al. [91] proposed Neural-Fly, which includes an online learning phase to overcome instabilities caused by wind effects. The approach includes offline and online learning. In offline learning, the DNN output is a set of basis functions that represent aerodynamic effects. The latter phase is an online adaptive control phase that learns to adapt the control system to new wind conditions rapidly and robustly. For offline learning, the domain adversarially invariant meta-learning (DAIML) algorithm is developed to learn aerodynamic effects under wind-invariant conditions. A stochastic gradient descent method is used in the DAIML algorithm for training. For online learning, a Kalman Filter-based adaptation algorithm estimates the wind-dependent linear coefficients. The position tracking error and the aerodynamic force prediction error terms are utilized in this estimation under wind-variant conditions. The online learning component provides an advantage of fast adaptation to wind variations. Also, the other advantage of the proposed approach is that it can be used to control several quadrotors without the need to pre-train each UAV. The proposed control algorithm was tested with the quadrotor in the Caltech Real Weather Wind Tunnel. Neural-Fly shows better performance than the nonlinear tracking controller found in [92], an $\mathcal{L}_1$ adaptive controller, and an incremental nonlinear dynamic inversion controller when the quadrotor is subjected to time-variant wind profiles.

Jia et al [93] provided a solution to the trajectory tracking problem by combining a fuzzy logic method, a radial basis function (RBF) NN, and a classical PID controller. The PID output and the current UAV position information are provided to the RBF NN as input values, and the network learns to adjust controller parameters. The fuzzy logic component selects the initial controller parameters, while the RBF NN adjusts them. Both are combined to create a new set of parameters for the PID controller. In the fuzzy logic component, a database created from expert knowledge is used to decide on the initial controller parameters. However, this dataset cannot be used in the RBF NN component since the adopted algorithm is an unsupervised learning method. However, the combination of both methods overcomes this limitation and provides online learning abilities. When the fuzzy logic system adjusts the PID gains, the RBF NN tweaks the PID parameters in real time to overcome PID weaknesses caused by environmental disturbances. When compared with PID and fuzzy-PID (FPID) controllers, this approach shows better performance for trajectory tracking.

Zhang et al. [94] investigated the problem of adaptive control and offered a real-time brain-inspired learning control (RBiLC) method as a solution. Three attitude angle errors are set as the input, and an NN provides the control parameter increment as the output. In the RBiLC method, a PID controller is used, and the controller parameter is updated in each interaction. A DRL method learns the controller parameter rate. The algorithm uses the Nesterov momentum technique for gradient descent. Controller stability and the convergence of the tracking error were demonstrated. The quadrotor takes off and reaches a 10-meter altitude, and then it hovers for 3 s in this position with the initial controller parameters. The learning algorithm is activated using a switch system, and the RBiLC algorithm updates PID gains in real time in an environment with wind disturbances. This approach learns controller parameters in 3 to 5 min, which is a much shorter time than in offline learning methods. This method shows significant improvement in stabilization in roll and pitch angles, but the performance is not the same in the yaw angle. Under wind disturbances, the method provides a shorter rise time and steady-state time for roll and pitch when compared to the classic PID controller.

Shiri et al. [95] studied the online path planning problem using NN-based opportunistic control. An online-trained NN learns to solve the Hamilton–Jacobian–Belmann (HJB) equation in real time. The opportunistic HJB, oHJB, control algorithm learns whether it will upload the control action (aHJB), which is the output of the NN, or the current online-trained NN (mHJB). A base station (BS) is utilized to handle the online learning algorithm. The UAV state is downloaded to the BS. First, the NN is trained online in the BS to solve the HJB in real time. Then, the output of the trained NN (the control action aHJB) is uploaded to the UAV. Secondly, the current online-trained NN (represented by mHJB) is uploaded to the UAV (instead of the aHJB), and the mHJB is fed the current states. Then, the UAV takes an action that is locally assessed by the uploaded mHJB in real time. The oHJB control provides the decision mechanism to switch from the aHJB to mHJB according to the connection between the multirotor UAV and the BS. Based on the oHJB, the UAV can keep taking actions using the last uploaded mHJB, even if it loses connection with the BS. Since the size of the trained NN model in the BS is larger than the size of the action space, a trade-off occurs between uploading delays and control robustness against poor channel conditions. The oHJB arrives at the targeted location in a shorter time than aHJB and mHJB. The aHJB and mHJB may fail to arrive at the desired location.

Table 4. Online learning papers.

Year	Paper	Task	Algorithm	Model-Free or Model-Based	Advantages	Compared with	Offline Part	Sim/Exp
2018	Yang et al. [86]	Navigation and synchronization	[86]	Model-free	Solves inhomogeneous algebraic Riccati equations online	Adaptive control approach in [87]	No	Sim
2018	Wang et al. [96]	Environment exploration	Data-driven approach based on Gaussian process	Model-free	Reduces possible crashes in the online learning phase	-	No	Sim
2019	Sarabakha and Kayacan [90]	Trajectory tracking	Back-propagation	Model-free	-	Offline-trained network, PID controller	Yes	Sim
2019	He et al. [97]	Agile mobility in a dynamic environment	StateRate	Model-free	Finely adjusts the prediction framework, and onboard sensor data are effectively used	Previous OPT, signal-to-noise rate (SNR), SampleRate, CHARM	Yes	Sim
2019	Wang et al. [98]	Robust control	DPG-IC	Model-free	Elimination of the steady error	PID controller, DDPG	Yes	Sim
2020	Shin et al. [89]	Speed optimization	SSD MobileNet	Model-free	Quicker object detection time	-	No	Sim
2020	Shiri et al. [95]	Path Planning	oHJB	Model-free	The algorithm keeps working even if UAV loses the connection with BS	aHJB, mHJB	No	Sim
2022	Jaiton et al. [88]	Speed optimization	Neural proactive control	Model-free	Computationally inexpensive	MPC	No	Exp
2023	O'Connell et al. [99]	Stabilization	DAIML	Model-free	Can control a wide range of quadrotors and not require pre-training for each UAV	Mellinger and Kumar [92], $\mathcal{L}_1$ adaptive controller, incremental nonlinear dynamic inversion controller	Yes	Exp
2023	Jia et al. [93]	Trajectory tracking	RFPID	Model-based	Strong learning ability	PID, Fuzzy-PID	No	Sim
2023	Zhang et al. [94]	Stabilization	RBiLC	Model-free	Significant improvement in stabilization in roll and pitch, but does not show the same performance in yaw	PID	No	Exp

Wang et al. [96] used a data-driven approach (Gaussian processes) to learn quadrotor models applied in partially unknown environments. Barrier certificates are learned and utilized to adjust the controller to not encroach on an unsafe region. A safe region is certified, and it is progressively spread with new data. Collecting more data points about system dynamics reduces uncertainty and maximizes the volume of the safe region. Discretizing the state space helps sample a finite number of points to affirm the barrier certificates; so, adaptive sampling decreases the number of required sampling points. Also, the state space is adaptively sampled by enhancing the Lipschitz continuity of the barrier certificates without taking any risk on safety guarantees. Sampling the most uncertain state in the safe set of the system boosts the learning efficiency during the exploration phase. A kernel function is utilized to decide on data relevance, and 300 data points are chosen in the recursive Gaussian process by eliminating irrelevant data points online. The approach reduces the possibility of an accident occurring during the online learning phase. The adaptive sampling strategy provides significantly better results in decreasing the required sample points. The quadrotor never violates safe and unsafe regions; it uses barrier certificates to regulate the learning algorithms. A high probability of safety guarantee is produced by a Gaussian process for the dynamic system, and this process learns unmodeled dynamics to help the quadrotor successfully stay within the safe region. No information is provided related to real-time applicability, but the authors report that they keep the Gaussian process interface time under 20 ms, and this allows for a suitable opportunity for online learning. When the tracking error of the learning-based controller is compared to the tracking error without Gaussian process inference, the tracking error is much smaller in the learning-based controller.

He et al. [97] solved the time-varying channel(s) problem on air-to-ground links caused by UAV mobility in dynamic environments. In the offline learning phase, the agent learns to minimize the prediction loss function in the prediction network and the evaluation loss function in the evaluation network. In online learning, the aim is to learn to minimize the difference between two networks. A state-optimized rate adaptation algorithm called StateRate was developed to solve this problem using the onboard UAV sensors. The evaluation and prediction networks are exploited in online training. The received signal strength indicator and the channel state information are used as inputs for both networks; the prediction network, additionally, uses the UAV's states as input. The output of the evaluation NN is used for supervision and compared with the output of the prediction network by using a fully connected layer. The StateRate algorithm accurately predicts the optimal rate. The rate prediction is handled as a multi-class classification problem using online learning. Retraining the pre-trained NN is not suitable, and StateRate with online learning provides an advantage to overcome this unsuitable situation. This method was applied and tested in a commercial quadrotor (DJI M100) and showed better performance than the best-known rate adaptation algorithms applied in UAVs with 2–6 m/s velocity.

Wang et al. [98] have worked on the robust control of a quadrotor using a DRL-based controller. The method includes both offline and online learning. During offline learning, an offline learning control policy is learned. The actor network learns the normalized thrusts produced by each rotor between 0 and 1. During online learning, the offline control policy continues to be optimized. In the offline phase, the Deterministic Policy Gradient-Integral Compensator (DPG-IC) algorithm is trained with random actor and critic NN weights in an episodic structure. In the online learning phase, the trained DPG-IC is used as the initial NN structure that is trained continuously. The offline DPG-IC also runs alongside the online algorithm. The system switches to the offline algorithm when the states are close to the safety limits, and the quadrotor goes back to the safe range. The aim of the online learning phase is to close the gap between the simplified model and the model with real flight dynamics. The main advantage of DPG-IG is that it removes the steady-state error. A well-tuned PID controller showed similar performance compared to the offline DPG-IC policy when the size of the quadrotor was increased from 0.12 m to 0.4 m, but for larger sizes, the PID controller was not stable, while the proposed method showed successful performance for sizes up to 1 m. The PID controller and the offline DPG-IC algorithm were also compared for different payloads. The payload of the quadrotor increased by 10% each step; the PID could not control the quadrotor with a 30% increased payload. On the other hand, the proposed offline learning method successfully controlled the quadrotor with up to a 50% payload increase. Then, the performance of the online learning policy was compared with that of the offline learning policy. The model was used with a 0.4 m radius and 20% increased payload. After only 200 online training steps, the performance was significantly increased.

In summary, since RL is built on the interaction between the agent and environment and is based on a reward system, RL algorithms are useful for online learning, and they are widely preferred for applications related to the control of multirotor UAVs.

5. Discussion and Conclusions

This survey provides summary results that are combined in four tables. The two main tables reflect offline and online learning techniques and algorithms. Both tables offer a clear picture of the publication year, the adopted method, the task/mission, and also what is being learned. Coupled with the provided information for each method, the reader acquires information about the applicability and implementability of each technique, as well as about the specific application the approach has been developed for. The other two tables offer specific details on RL approaches and on the value-function-based and policy-search-based methods. Overall, this survey provides a comprehensive overview of the evolving landscape of multirotor UAV navigation and control, particularly focusing on the integration of learning-based algorithms over the past decade.

Online learning allows for an agent to continuously learn and update its decision-making process using real-time data, enhancing its capabilities to complete different missions. In online learning, as opposed to offline learning, the agent has the ability to learn directly from problem instances. In this survey, 11 papers following online learning for multirotor UAV control demonstrate efficacy in tasks such as trajectory tracking, speed adaptation, obstacle avoidance, and path planning. The used techniques include neural network-based controllers, RL algorithms, and adaptive control methods. These methods offer advantages such as computational efficiency, real-time adaptability, and robust performance in dynamic environments or under disturbances, like wind gusts.

A common characteristic of online learning techniques is that all methods are model-free, except for the proposed approach in [93].

Four reviewed articles include both offline and online learning phases, while seven articles have only online learning phases. The online learning phase (used after offline learning) provides an opportunity to learn how to compensate for differences between basic and complex UAV models and for environmental disturbances or to reduce errors. Applying online learning directly without an offline learning phase helps considerably reduce the training time. For example, in [88], only 3–4 trials are required for learning, while in [89], learning occurs in every attempt, reducing mission completion time. However, an important observation in online learning algorithms is that NNs cannot be used alone; they are always combined with auxiliary components, such as conventional controllers, fuzzy logic systems, and optimization or adaptation algorithms, to decrease computational cost and learning time or to increase adaptation abilities. In addition, the BS may be used to increase the computational capacity, and 5G technology helps transfer the trained agent or output of the NNs to the multirotor UAVs, as shown in [95]. Reducing the number of neurons also offers advantages because of the reduction in the computational cost [88].

Learning algorithms, in general, have witnessed wide applicability in diverse applications. They can be used in combination with other approaches, too. However, one of their limitations is proof of stability. There is limited literature, which has been reviewed, where proof of stability is shown [86,94,96,99].

Overall, online learning emerges as a powerful tool for enhancing UAV control and navigation in complex scenarios, but it still includes significant limitations, as mentioned before. Given that, eventually, there will be ‘almost infinite computational power’ that will require ‘almost zero computational time’ to return results, this survey paper offers a starting point for subsequent studies on what is hard real-time implementable.

Author Contributions: This work is part of S.S.’s PhD dissertation. S.S. is the main contributor in the paper, with M.J.R. and K.P.V. reading, editing and recommending additions and modifications to the contents. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the Ministry of National Education of the Republic of Turkey on behalf of Istanbul Medeniyet University and the D. F. Ritchie School of Engineering and Computer Science, University of Denver, CO 80208.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

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

Abbreviations

The following abbreviations are used in this manuscript:

UAV	Unmanned aerial vehicle
RL	Reinforcement learning
DRL	Deep reinforcement learning
SARSA	State–Action–Reward–State–Action
DDPG	Deep deterministic policy gradient
ANN	Artificial neural network
ML	Machine learning
DL	Deep learning
MDP	Markov decision process
AI	Artificial Intelligence
DNN	Deep neural network
NN	Neural network
TLD	Tracking–Learning–Detection
GCS	Ground control station
CNN	Convolutional neural network
VIO	Visual-inertial odometry
RMLP	Recurrent multilayer perceptron
LSTM	Long Short-Term Memory
PID	Proportional–Integral–Derivative
LQR	Linear–Quadratic Regulator
RGB	Red–blue–green
MAE	Mean Absolute Error
SSD	Single-Shot Detection
LOS	Line-Of-Sight
MAV	Micro aerial vehicle
DSO	Direct sparse odometry
LSTMCNN	LSTM Layers interleaved with convolutional 1D layers
CLSTM	Convolutional 1D Layers cascaded with LSTM layers
DP	Dynamic programming
MC	Monte Carlo
TD	Temporal Difference
DQN	Deep Q-Network
AR	Augmented reality
GAT	Graph attention network
FANAT	Flying ad hoc network

GAT-FANET	GAT-based FANET
DRGN	Deep Recurrent Graph Network
SDRGN	Soft deep recurrent graph network
GRU	Gated recurrent unit
MAAC	Multi-actor attention critic
DGN	Graph Convolutional Reinforcement Learning
AMLQ	Adaptive multi-level quantization
RRT	Rapidly exploring random tree
RSS	Received signal strength
HMM	Hidden Markov model
DTW	Dynamic time warping
DDQN	Double DQN
LSPI	Least-Square Policy Iteration
ADP	Approximate dynamic programming
SOL	Structured online learning-based algorithm
PWM	Pulse-width modulation
IBVS	Image-based Visual Servoing
PBVS	Position-based Visual Servoing
PPO	Proximal Policy Optimization
PPO-IC	Proximal Policy Optimization-Integral Compensator
TD3	Twin-Delay Deep Deterministic Gradient
SAC	Soft Action–Critic
PLATO	Policy Learning using Adaptive Trajectory Optimization
MPC	Model predictive control
UKF	Unscented Kalman Filter
PF	Particle Filter
TRPO	Trust Region Policy Optimization
PILCO	Probabilistic inference for learning control
NLGL	Nonlinear Guidance Law
POMDP	Partially observable Markov decision process
RDPG	Recurrent deterministic policy gradient algorithm
DeFRA	DRL-based flight resource allocation framework
DQN-FRA	DQNs-based Flight Resource Allocation Policy
CAWS	Channel-Aware Waypoint Selection
PTRS	Planned Trajectory Random Scheduling
DDPG-MC	DDPG-based Movement Control
PR	Policy relief
SW	Significance weighting
TRPO-gae	Trust Region Policy Optimization with a generalized advantage estimator
UGV	Unmanned Ground Vehicle
meta-TD3	Meta twin delay deep deterministic policy gradient
CdRL	Consciousness-driven reinforcement learning
MAPPO	Multi-agent PPO
HAPPO	Heterogeneous-agent PPO
MADDPG	Multi-agent DDPG
ARE	Algebraic Riccati equation
FLS	Fuzzy logic system
DAIML	Domain adversarially invariant meta-learning
RBF	Radial basis function
FPID	Fuzzy-PID
RBiLC	Real-time brain-inspired learning control
HJB	Hamilton–Jacobian–Belmann equation
BS	Base station
DPG-IC	Deterministic Policy Gradient-Integral Compensator

References

1. Martinez, C.; Sampedro, C.; Chauhan, A.; Campoy, P. Towards autonomous detection and tracking of electric towers for aerial power line inspection. In Proceedings of the 2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, USA, 27–30 May 2014; pp. 284–295.
2. Ren, H.; Zhao, Y.; Xiao, W.; Hu, Z. A review of UAV monitoring in mining areas: Current status and future perspectives. *Int. J. Coal Sci. Technol.* **2019**, *6*, 320–333. [CrossRef]
3. Olivares-Mendez, M.A.; Fu, C.; Ludivig, P.; Bissyandé, T.F.; Kannan, S.; Zurad, M.; Annaiyan, A.; Voos, H.; Campoy, P. Towards an autonomous vision-based unmanned aerial system against wildlife poachers. *Sensors* **2015**, *15*, 31362–31391. [CrossRef] [PubMed]
4. Bassoli, R.; Sacchi, C.; Granelli, F.; Ashkenazi, I. A virtualized border control system based on UAVs: Design and energy efficiency considerations. In Proceedings of the 2019 IEEE Aerospace Conference, Big Sky, MT, USA, 2–9 March 2019; pp. 1–11.
5. Carrio, A.; Pestana, J.; Sanchez-Lopez, J.L.; Suarez-Fernandez, R.; Campoy, P.; Tendero, R.; García-De-Viedma, M.; González-Rodrigo, B.; Bonatti, J.; Rejas-Ayuga, J.G.; et al. UBRISTES: UAV-based building rehabilitation with visible and thermal infrared remote sensing. In Proceedings of the Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Lisbon, Portugal, 19–21 November 2015; Springer: Berlin/Heidelberg, Germany, 2016; Volume 1, pp. 245–256.
6. Li, L.; Fan, Y.; Huang, X.; Tian, L. Real-time UAV weed scout for selective weed control by adaptive robust control and machine learning algorithm. In Proceedings of the 2016 ASABE Annual International Meeting. American Society of Agricultural and Biological Engineers, Orlando, FL, USA, 17 July–20 July 2016; p. 1.
7. Carrio, A.; Sampedro, C.; Rodriguez-Ramos, A.; Campoy, P. A review of deep learning methods and applications for unmanned aerial vehicles. *J. Sens.* **2017**, *2017*, 3296874. [CrossRef]
8. Polydoros, A.S.; Nalpantidis, L. Survey of model-based reinforcement learning: Applications on robotics. *J. Intell. Robot. Syst.* **2017**, *86*, 153–173. [CrossRef]
9. Choi, S.Y.; Cha, D. Unmanned aerial vehicles using machine learning for autonomous flight; state-of-the-art. *Adv. Robot.* **2019**, *33*, 265–277. [CrossRef]
10. Azar, A.T.; Koubaa, A.; Ali Mohamed, N.; Ibrahim, H.A.; Ibrahim, Z.F.; Kazim, M.; Ammar, A.; Benjdira, B.; Khamis, A.M.; Hameed, I.A.; et al. Drone deep reinforcement learning: A review. *Electronics* **2021**, *10*, 999. [CrossRef]
11. Brunke, L.; Greeff, M.; Hall, A.W.; Yuan, Z.; Zhou, S.; Panerati, J.; Schoellig, A.P. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annu. Rev. Control. Robot. Auton. Syst.* **2022**, *5*, 411–444. [CrossRef]
12. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
13. Levine, S.; Kumar, A.; Tucker, G.; Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv* **2020**, arXiv:2005.01643.
14. Bartak, R.; Vykovskỳ, A. Any object tracking and following by a flying drone. In Proceedings of the 2015 Fourteenth Mexican International Conference on Artificial Intelligence (MICAI), Cuernavaca, Mexico, 25–21 October 2015; pp. 35–41.
15. Edhah, S.; Mohamed, S.; Rehan, A.; AlDhaheri, M.; AlKhaja, A.; Zweiri, Y. Deep Learning Based Neural Network Controller for Quad Copter: Application to Hovering Mode. In Proceedings of the 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 19–21 November 2019; pp. 1–5.
16. Xu, Y.; Liu, Z.; Wang, X. Monocular vision based autonomous landing of quadrotor through deep reinforcement learning. In Proceedings of the 2018 37th Chinese Control Conference (CCC), Wuhan, China, 25–27 July 2018; pp. 10014–10019.
17. Rodriguez-Ramos, A.; Sampedro, C.; Bavle, H.; De La Puente, P.; Campoy, P. A deep reinforcement learning strategy for UAV autonomous landing on a moving platform. *J. Intell. Robot. Syst.* **2019**, *93*, 351–366. [CrossRef]
18. Yoo, J.; Jang, D.; Kim, H.J.; Johansson, K.H. Hybrid reinforcement learning control for a micro quadrotor flight. *IEEE Control Syst. Lett.* **2020**, *5*, 505–510. [CrossRef]
19. Hoi, S.C.; Sahoo, D.; Lu, J.; Zhao, P. Online learning: A comprehensive survey. *Neurocomputing* **2021**, *459*, 249–289. [CrossRef]
20. Giusti, A.; Guzzi, J.; Cireşan, D.C.; He, F.L.; Rodríguez, J.P.; Fontana, F.; Faessler, M.; Forster, C.; Schmidhuber, J.; Di Caro, G.; et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robot. Autom. Lett.* **2015**, *1*, 661–667. [CrossRef]
21. Kaufmann, E.; Loquercio, A.; Ranftl, R.; Dosovitskiy, A.; Koltun, V.; Scaramuzza, D. Deep drone racing: Learning agile flight in dynamic environments. In Proceedings of the Conference on Robot Learning, Zürich, Switzerland, 29–31 October 2018; pp. 133–145.
22. Janousek, J.; Marcon, P.; Klouda, J.; Pokorny, J.; Raichl, P.; Siruckova, A. Deep Neural Network for Precision Landing and Variable Flight Planning of Autonomous UAV. In Proceedings of the 2021 Photonics & Electromagnetics Research Symposium (PIERS), Hangzhou, China, 21–25 November 2021; pp. 2243–2247.
23. Vladov, S.; Shmelov, Y.; Yakovliev, R.; Khebda, A.; Brusakova, O. Modified Neural Network Method for Stabilizing Multi-Rotor Unmanned Aerial Vehicles. In Proceedings of the 7th International Conference on Computational Linguistics and Intelligent Systems, Kharkiv, Ukraine, 20–21 April 2023.
24. Kim, D.K.; Chen, T. Deep neural network for real-time autonomous indoor navigation. *arXiv* **2015**, arXiv:1511.04668.
25. Li, Q.; Qian, J.; Zhu, Z.; Bao, X.; Helwa, M.K.; Schoellig, A.P. Deep neural networks for improved, impromptu trajectory tracking of quadrotors. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 5183–5189.

26. Smolyanskiy, N.; Kamenev, A.; Smith, J.; Birchfield, S. Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 4241–4247.
27. Jung, S.; Hwang, S.; Shin, H.; Shim, D.H. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robot. Autom. Lett.* **2018**, *3*, 2539–2544. [CrossRef]
28. Loquercio, A.; Maqueda, A.I.; Del-Blanco, C.R.; Scaramuzza, D. Dronet: Learning to fly by driving. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1088–1095. [CrossRef]
29. Mantegazza, D.; Guzzi, J.; Gambardella, L.M.; Giusti, A. Vision-based control of a quadrotor in user proximity: Mediated vs end-to-end learning approaches. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6489–6495.
30. Cardenas, J.A.; Carrero, U.E.; Camacho, E.C.; Calderon, J.M. Intelligent Position Controller for Unmanned Aerial Vehicles (UAV) Based on Supervised Deep Learning. *Machines* **2023**, *11*, 606. [CrossRef]
31. Imanberdiyev, N.; Fu, C.; Kayacan, E.; Chen, I.M. Autonomous navigation of UAV by using real-time model-based reinforcement learning. In Proceedings of the 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), Phuket, Thailand, 13–15 November 2016; pp. 1–6.
32. Polvara, R.; Patacchiola, M.; Sharma, S.; Wan, J.; Manning, A.; Sutton, R.; Cangelosi, A. Autonomous quadrotor landing using deep reinforcement learning. *arXiv* **2017**, arXiv:1709.03339.
33. Choi, S.; Kim, S.; Jin Kim, H. Inverse reinforcement learning control for trajectory tracking of a multirotor UAV. *Int. J. Control. Autom. Syst.* **2017**, *15*, 1826–1834. [CrossRef]
34. Kahn, G.; Zhang, T.; Levine, S.; Abbeel, P. Plato: Policy learning using adaptive trajectory optimization. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3342–3349.
35. Hwangbo, J.; Sa, I.; Siegwart, R.; Hutter, M. Control of a quadrotor with reinforcement learning. *IEEE Robot. Autom. Lett.* **2017**, *2*, 2096–2103. [CrossRef]
36. Lee, S.; Shim, T.; Kim, S.; Park, J.; Hong, K.; Bang, H. Vision-based autonomous landing of a multi-copter unmanned aerial vehicle using reinforcement learning. In Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, USA, 12–15 June 2018; pp. 108–114.
37. Vankadari, M.B.; Das, K.; Shinde, C.; Kumar, S. A reinforcement learning approach for autonomous control and landing of a quadrotor. In Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, USA, 12–15 June 2018; pp. 676–683.
38. Kersandt, K.; Muñoz, G.; Barrado, C. Self-training by reinforcement learning for full-autonomous drones of the future. In Proceedings of the 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), London, UK, 23–27 September 2018; pp. 1–10.
39. Pham, H.X.; La, H.M.; Feil-Seifer, D.; Van Nguyen, L. Reinforcement learning for autonomous uav navigation using function approximation. In Proceedings of the 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Philadelphia, PA, USA, 6–8 August 2018; pp. 1–6.
40. Liu, H.; Zhao, W.; Lewis, F.L.; Jiang, Z.P.; Modares, H. Attitude synchronization for multiple quadrotors using reinforcement learning. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 2480–2483.
41. Lambert, N.O.; Drew, D.S.; Yaconelli, J.; Levine, S.; Calandra, R.; Pister, K.S. Low-level control of a quadrotor with deep model-based reinforcement learning. *IEEE Robot. Autom. Lett.* **2019**, *4*, 4224–4230. [CrossRef]
42. Manukyan, A.; Olivares-Mendez, M.A.; Geist, M.; Voos, H. Deep Reinforcement Learning-based Continuous Control for Multicopter Systems. In Proceedings of the 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), Paris, France, 23–26 April 2019; pp. 1876–1881.
43. Srivastava, R.; Lima, R.; Das, K.; Maity, A. Least square policy iteration for ibvs based dynamic target tracking. In Proceedings of the 2019 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 11–14 June 2019; pp. 1089–1098.
44. Wu, C.; Ju, B.; Wu, Y.; Lin, X.; Xiong, N.; Xu, G.; Li, H.; Liang, X. UAV autonomous target search based on deep reinforcement learning in complex disaster scene. *IEEE Access* **2019**, *7*, 117227–117245. [CrossRef]
45. Wang, C.; Wang, J.; Shen, Y.; Zhang, X. Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach. *IEEE Trans. Veh. Technol.* **2019**, *68*, 2124–2136. [CrossRef]
46. Zeng, Y.; Xu, X. Path design for cellular-connected UAV with reinforcement learning. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Big Island, HI, USA, 9–13 December 2019; pp. 1–6.
47. Rubí, B.; Morcego, B.; Pérez, R. A Deep Reinforcement Learning Approach for Path Following on a Quadrotor. In Proceedings of the 2020 European Control Conference (ECC), Saint Petersburg, Russia, 12–15 May 2020; pp. 1092–1098.
48. Pi, C.H.; Hu, K.C.; Cheng, S.; Wu, I.C. Low-level autonomous control and tracking of quadrotor using reinforcement learning. *Control Eng. Pract.* **2020**, *95*, 104222. [CrossRef]
49. Zhao, W.; Liu, H.; Lewis, F.L. Robust formation control for cooperative underactuated quadrotors via reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4577–4587. [CrossRef] [PubMed]
50. Guerra, A.; Guidi, F.; Dardari, D.; Djurić, P.M. Reinforcement learning for UAV autonomous navigation, mapping and target detection. In Proceedings of the 2020 IEEE/ION Position, Location and Navigation Symposium (PLANS), Portland, OR, USA, 20–23 April 2020; pp. 1004–1013.

51. Li, B.; Gan, Z.; Chen, D.; Sergey Aleksandrovich, D. UAV maneuvering target tracking in uncertain environments based on deep reinforcement learning and meta-learning. *Remote Sens.* **2020**, *12*, 3789. [CrossRef]
52. Kulkarni, S.; Chaphekar, V.; Chowdhury, M.M.U.; Erden, F.; Guvenc, I. UAV aided search and rescue operation using reinforcement learning. In Proceedings of the 2020 SoutheastCon, Raleigh, NC, USA, 28–29 March 2020; Volume 2, pp. 1–8.
53. Hu, H.; Wang, Q.l. Proximal policy optimization with an integral compensator for quadrotor control. *Front. Inf. Technol. Electron. Eng.* **2020**, *21*, 777–795. [CrossRef]
54. Kooi, J.E.; Babuška, R. Inclined Quadrotor Landing using Deep Reinforcement Learning. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 2361–2368.
55. Rubí, B.; Morcego, B.; Pérez, R. Deep reinforcement learning for quadrotor path following with adaptive velocity. *Auton. Robot.* **2021**, *45*, 119–134. [CrossRef]
56. Bhan, L.; Quinones-Grueiro, M.; Biswas, G. Fault Tolerant Control combining Reinforcement Learning and Model-based Control. In Proceedings of the 2021 5th International Conference on Control and Fault-Tolerant Systems (SysTol), Saint-Raphael, France, 29 September–1 October2021; pp. 31–36.
57. Li, X.; Zhang, J.; Han, J. Trajectory planning of load transportation with multi-quadrotors based on reinforcement learning algorithm. *Aerosp. Sci. Technol.* **2021**, *116*, 106887. [CrossRef]
58. Jiang, Z.; Song, G. A Deep Reinforcement Learning Strategy for UAV Autonomous Landing on a Platform. *arXiv* **2022**, arXiv:2209.02954.
59. Abo Mosali, N.; Shamsudin, S.S.; Mostafa, S.A.; Alfandi, O.; Omar, R.; Al-Fadhali, N.; Mohammed, M.A.; Malik, R.; Jaber, M.M.; Saif, A. An Adaptive Multi-Level Quantization-Based Reinforcement Learning Model for Enhancing UAV Landing on Moving Targets. *Sustainability* **2022**, *14*, 8825. [CrossRef]
60. Panetsos, F.; Karras, G.C.; Kyriakopoulos, K.J. A deep reinforcement learning motion control strategy of a multi-rotor uav for payload transportation with minimum swing. In Proceedings of the 2022 30th Mediterranean Conference on Control and Automation (MED), Vouliagmeni, Greece, 28 June–1 July 2022; pp. 368–374.
61. Ye, Z.; Wang, K.; Chen, Y.; Jiang, X.; Song, G. Multi-UAV Navigation for Partially Observable Communication Coverage by Graph Reinforcement Learning. *IEEE Trans. Mob. Comput.* **2022**, *22*, 4056–4069. [CrossRef]
62. Wang, X.; Ye, X. Consciousness-driven reinforcement learning: An online learning control framework. *Int. J. Intell. Syst.* **2022**, *37*, 770–798. [CrossRef]
63. Farsi, M.; Liu, J. Structured online learning for low-level control of quadrotors. In Proceedings of the 2022 American Control Conference (ACC), Atlanta, GA, USA, 8–10 June 2022; pp. 1242–1247.
64. Xia, K.; Huang, Y.; Zou, Y.; Zuo, Z. Reinforcement Learning Control for Moving Target Landing of VTOL UAVs with Motion Constraints. *IEEE Trans. Ind. Electron.* **2023**. [CrossRef]
65. Ma, B.; Liu, Z.; Dang, Q.; Zhao, W.; Wang, J.; Cheng, Y.; Yuan, Z. Deep Reinforcement Learning of UAV Tracking Control Under Wind Disturbances Environments. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 2510913. [CrossRef]
66. Castro, G.G.d.; Berger, G.S.; Cantieri, A.; Teixeira, M.; Lima, J.; Pereira, A.I.; Pinto, M.F. Adaptive Path Planning for Fusing Rapidly Exploring Random Trees and Deep Reinforcement Learning in an Agriculture Dynamic Environment UAVs. *Agriculture* **2023**, *13*, 354. [CrossRef]
67. Mitakidis, A.; Aspragkathos, S.N.; Panetsos, F.; Karras, G.C.; Kyriakopoulos, K.J. A Deep Reinforcement Learning Visual Servoing Control Strategy for Target Tracking Using a Multirotor UAV. In Proceedings of the 2023 9th International Conference on Automation, Robotics and Applications (ICARA), Abu Dhabi, United Arab Emirates, 10–12 February 2023; pp. 219–224.
68. Shurrab, M.; Mizouni, R.; Singh, S.; Otrok, H. Reinforcement learning framework for UAV-based target localization applications. *Internet Things* **2023**, *23*, 100867. [CrossRef]
69. Santana, P.; Correia, L.; Mendonça, R.; Alves, N.; Barata, J. Tracking natural trails with swarm-based visual saliency. *J. Field Robot.* **2013**, *30*, 64–86. [CrossRef]
70. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
71. Caffe | Model Zoo. Available online: http://caffe.berkeleyvision.org/model_zoo.html (accessed on 11 October 2022).
72. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
73. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
74. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
75. Li, K.; Ni, W.; Dressler, F. LSTM-characterized Deep Reinforcement Learning for Continuous Flight Control and Resource Allocation in UAV-assisted Sensor Network. *IEEE Internet Things J.* **2021**, *9*, 4179–4189. [CrossRef]
76. Bilgin, E. *Mastering Reinforcement Learning with Python: Build Next-Generation, Self-Learning Models Using Reinforcement Learning Techniques and Best Practices*; Packt Publishing Ltd.: Birmingham, UK, 2020.
77. Lapan, M. *Deep Reinforcement Learning Hands-On: Apply Modern RL Methods to Practical Problems of Chatbots, Robotics, Discrete Optimization, Web Automation, and More*; Packt Publishing Ltd.: Birmingham, UK, 2020.

78. Lapan, M. *Deep Reinforcement Learning Hands-On: Apply Modern RL Methods, with Deep Q-Networks, Value Iteration, Policy Gradients, TRPO, AlphaGo Zero and More*; Packt Publishing Ltd.: Birmingham, UK, 2018.
79. Polvara, R.; Patacchiola, M.; Hanheide, M.; Neumann, G. Sim-to-Real quadrotor landing via sequential deep Q-Networks and domain randomization. *Robotics* **2020**, *9*, 8. [CrossRef]
80. Farsi, M.; Liu, J. Structured online learning-based control of continuous-time nonlinear systems. *IFAC-PapersOnLine* **2020**, *53*, 8142–8149. [CrossRef]
81. Kanellakis, C.; Nikolakopoulos, G. Survey on computer vision for UAVs: Current developments and trends. *J. Intell. Robot. Syst.* **2017**, *87*, 141–168. [CrossRef]
82. Stevens, B.L.; Lewis, F.L.; Johnson, E.N. *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*; John Wiley & Sons: Hoboken, NJ, USA, 2015.
83. Xu, B.; Gao, F.; Yu, C.; Zhang, R.; Wu, Y.; Wang, Y. Omnidrones: An efficient and flexible platform for reinforcement learning in drone control. *IEEE Robot. Autom. Lett.* **2024**, *9*, 2838–2844. [CrossRef]
84. Srinivasan, D. *Innovations in Multi-Agent Systems and Application—1*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 310.
85. Van Otterlo, M.; Wiering, M. Reinforcement learning and markov decision processes. In *Reinforcement Learning: State-of-the-Art*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 3–42.
86. Yang, Y.; Modares, H.; Wunsch, D.C.; Yin, Y. Leader–follower output synchronization of linear heterogeneous systems with active leader using reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 2139–2153. [CrossRef]
87. Das, A.; Lewis, F.L. Distributed adaptive control for synchronization of unknown nonlinear networked systems. *Automatica* **2010**, *46*, 2014–2021. [CrossRef]
88. Jaiton, V.; Rothomphiwat, K.; Ebeid, E.; Manoonpong, P. Neural Control and Online Learning for Speed Adaptation of Unmanned Aerial Vehicles. *Front. Neural Circuits* **2022**, *16*, 839361. [CrossRef]
89. Shin, S.; Kang, Y.; Kim, Y.G. Evolution algorithm and online learning for racing drone. In Proceedings of the NeurIPS 2019 Competition and Demonstration Track, Vancouver, BC, Canada, 10–12 December 2020; pp. 100–109.
90. Sarabakha, A.; Kayacan, E. Online deep learning for improved trajectory tracking of unmanned aerial vehicles using expert knowledge. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 7727–7733.
91. O'Connell, M.; Shi, G.; Shi, X.; Azizzadenesheli, K.; Anandkumar, A.; Yue, Y.; Chung, S.J. Neural-fly enables rapid learning for agile flight in strong winds. *Sci. Robot.* **2022**, *7*, eabm6597. [CrossRef] [PubMed]
92. Mellinger, D.; Kumar, V. Minimum snap trajectory generation and control for quadrotors. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2520–2525.
93. Jia, K.; Lin, S.; Du, Y.; Zou, C.; Lu, M. Research on route tracking controller of Quadrotor UAV based on fuzzy logic and RBF neural network. *IEEE Access* **2023**, *11*, 111433–111447. [CrossRef]
94. Zhang, Y.; Yang, Y.; Chen, W.; Yang, H. Realtime Brain-Inspired Adaptive Learning Control for Nonlinear Systems with Configuration Uncertainties. *IEEE Trans. Autom. Sci. Eng.* **2023**. [CrossRef]
95. Shiri, H.; Park, J.; Bennis, M. Remote UAV online path planning via neural network-based opportunistic control. *IEEE Wirel. Commun. Lett.* **2020**, *9*, 861–865. [CrossRef]
96. Wang, L.; Theodorou, E.A.; Egerstedt, M. Safe learning of quadrotor dynamics using barrier certificates. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 2460–2465.
97. He, S.; Wang, W.; Yang, H.; Cao, Y.; Jiang, T.; Zhang, Q. State-aware rate adaptation for UAVs by incorporating on-board sensors. *IEEE Trans. Veh. Technol.* **2019**, *69*, 488–496. [CrossRef]
98. Wang, Y.; Sun, J.; He, H.; Sun, C. Deterministic policy gradient with integral compensator for robust quadrotor control. *IEEE Trans. Syst. Man, Cybern. Syst.* **2019**, *50*, 3713–3725. [CrossRef]
99. O'Connell, M.; Shi, G.; Shi, X.; Azizzadenesheli, K.; Anandkumar, A.; Yue, Y.; Chung, S.J. Pretraining Neural-Networks with Neural-Fly for Rapid Online Learning. In Proceedings of the ICRA2023 Workshop on Pretraining for Robotics (PT4R), London, UK, 29 May 2023.

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.