electronics

MDPI

Article

Trajectory Control of Quadrotors via Spiking Neural Networks

Yesim Oniz

Department of Mechatronics Engineering, Istanbul Bilgi University, Istanbul 34060, Türkiye; yesim.oniz@bilgi.edu.tr

Abstract: In this study, a novel control scheme based on spiking neural networks (SNNs) has been proposed to accomplish the trajectory tracking of quadrotor unmanned aerial vehicles (UAVs). The update rules for the network parameters have been derived using the Lyapunov stability theorem. Three different trajectories have been utilized in the simulated and experimental studies to verify the efficacy of the proposed control scheme. The acquired results have been compared with the responses obtained for proportional–integral–derivative (PID) and traditional neural network controllers. Simulated and experimental studies demonstrate that the proposed SNN-based controller is capable of providing better tracking accuracy and robust system response in the presence of disturbing factors.

Keywords: rotary wing unmanned aerial vehicle; spiking neural networks; trajectory tracking

check for updates

Citation: Oniz, Y. Trajectory Control of Quadrotors via Spiking Neural Networks. *Electronics* **2024**, *13*, 3319. https://doi.org/10.3390/electronics13163319

Academic Editors: Luca Patanè, Andon Venelinov Topalov and Sevil Ahmed

Received: 9 June 2024
Revised: 1 August 2024
Accepted: 19 August 2024
Published: 21 August 2024

Copyright: © 2024 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The last century has witnessed tremendous advances in aircraft technology. Since the first manned flight in 1903, aircraft designs have gradually evolved, making it possible today to design aircraft that can carry large cargo, exhibit high maneuverability, and reach considerable speeds and altitudes. From the second half of the 20th century, unmanned aerial vehicles (UAVs) have been developed and used for various military purposes, and the interest in these aircraft has been increasing steadily due to the advantages they provide in complex and uncertain environments. UAVs are especially preferred over full-scale manned aircraft for missions where physical conditions are challenging or hazardous. In the last two decades, the development of affordable, stable, and high-performance electronic components and systems and the easy availability of the necessary materials have made the research and production of unmanned aerial vehicles more widespread. In addition to their use in military fields, they have been preferred in different applications including search and rescue missions, aerial imaging, border security, mapping, topography, and fire tracking [1–5].

In this study, the trajectory tracking problem of a quadrotor, which is a rotary-wing type UAV with four rotors, has been addressed. Quadrotors are extensively used in different areas due to their small size, simple mechanical structure, vertical takeoff and landing properties, and agility. Despite the advantages they offer, control of the altitude and attitude of this type of UAV is a rather challenging task due to coupled, nonlinear dynamics of the UAV along with the uncertainties and disturbing factors present in the operating environment [6]. In addition, simultaneous control of multiple rotors, uncertainties, noise associated with the sensor data, and varying weather conditions throughout the operation further complicate the control of UAVs. To alleviate these issues, two different approaches are generally adopted by researchers. The use of model-based controllers, which require an accurate mathematical model of the system, constitutes the first approach, while in the other approach, this requirement has been overcome by adopting model-free controllers.

Model-based linear controllers such as proportional–integral–derivative (PID) controllers and linear quadratic regulators (LQR) have achieved considerable success in controlling UAVs. In [7], the performance of a rotary wing UAV using PID controllers for altitude and attitude control has been tested for different trajectories requiring hard maneuvers.

In [8], both LQR and PID control techniques have been used and the authors reported "average" results. In [9], LQR has been used in combination with an optimal trajectory generator. A series of simulation studies have been performed that required changes in the reference trajectory during the mission. Model predictive control (MPC) is another model-based control method that addresses the trajectory tracking problem of UAVs. Until recently, real-time applications of MPC were limited due to its high computational requirements, but the advent of high-performance processors has enabled the use of such controllers in aeronautical engineering, where the sampling time of the system must be fast enough. In [10], various operating points for a rotary wing UAV have been first identified, and the dynamics of the system have been linearized for these points to obtain a set of linear equations. For each linear region, MPC provided the optimal control signal, while at the same time allowing smooth transitions from one linear region to another. In [11], a linear predictor has been utilized to predict the future values of the system to reduce the computational burden of the MPC controller.

In aircraft engineering, the creation of a highly accurate mathematical model of the system requires the collection of wind tunnel data by performing a series of flight tests and then adapting the model to the data. This process is laborious, and the resulting model may have uncertainty in the parameters due to the assumptions and simplifications used as well as performance uncertainty due to material properties and manufacturing tolerances [12–15]. The uncertainties resulting from different components and sources have different distribution characteristics. In [16], uncertainty quantification has been performed to better understand the performance variations caused by the uncertainties in the components. It is argued that the uncertainties related to the battery and rotor diameter have important effects on the hovering time and successful completion/failure of the mission. In addition, the raw data obtained from the sensors is usually uncertain and noisy [17], as the outputs of the sensors depend on the temperature, sampling frequency, humidity, power supply variations, aging, and electromagnetic interference [18,19]. Furthermore, transmission of sensor signals over long distances or wireless means may also cause corruption in the data. Inertial Measurement Unit (IMU) sensors are the most widely used sensors in quadrotor systems. This sensor encompasses accelerometers and gyroscopes to measure acceleration and rotation, which are used to obtain the position data of the quadrotor. Usually, IMU sensors are set to operate at high frequencies, resulting in noisy readings [20]. Additionally, the parameters in the operating environment of the quadrotor, such as temperature, pressure, and precipitation, frequently change throughout the course of the flight, resulting in errors in sensor readings [21]. To cope with uncertain and noisy sensor signals, different approaches can be adopted, which include the use of dynamic, low-pass, or Kalman filters [22,23]. Intelligent control systems such as fuzzy logic systems and artificial neural networks pose another viable solution. In these model-free control schemes, instead of using sensor data and the results of experiments to build a mathematical model of the physical system, these are directly utilized in the design of the controller, which eliminates the need for a highly accurate model of the system and facilitates controller design in situations involving uncertainties and nonlinearities.

In artificial neural networks (ANNs), the learning mechanism of the human brain is mimicked by a network of artificial neurons. Using the interactions between neurons, the network adapts the weights of the network to produce the desired outputs. This learning capability of the ANNs enables their extensive use for pattern and image recognition, data classification, time series forecasting, and control applications. ANNs have been preferred by many researchers in the design of controllers for quadrotors [24]. In [25], a neural network combined with state-dependent Riccati Equations [26] has been employed in the position and velocity control of a small quadrotor. In [27], it is assumed that the the dynamical model of the system is uncertain, and linear and angular velocities of the rotary wing UAV cannot be measured. An ANN has been employed to estimate the velocity vector, which has been utilized to control the position and orientation of the UAV. In [28], identification and control of the UAV have been realized with adaptive neural networks.

First, a model of the UAV has been generated by an ANN, and then a radial basis function (RBF) network controller has been developed for this model. Ref. [29] proposes an approach where the translational dynamics of the UAV have been derived using a feedforward neural network, which has been then used as the prediction model in MPC.

Multilayer perceptron (MLP) neural networks have been the predominant network structure for many years. They are typical examples of the second generation of neural networks, in which the hard-nonlinearity activation function of the previous generation has been replaced with a continuous activation function, allowing ANNs to be used in systems, where the inputs and outputs are analog. Spiking neural networks (SNNs), also known as third-generation neural networks, have recently grown in popularity due to their ability to increase biological realism by utilizing temporal information of individual spikes. Biological neurons use short, momentary voltage spikes to transmit information. These signals are often referred to as spikes. In the earlier neuron models, output values typically lie between 0 and 1, which can be thought of as the average firing frequency of the neuron. It was generally thought that the only information transmitted between two biological neurons was the firing frequency. Numerous neurobiological studies have shown that information is encoded in the precise timing of spikes, not in their firing rate [30,31]. SNN models, like biological neurons, encode the data to be transmitted between neurons in the temporal information of the spikes, with all spikes having the same magnitude. This makes SNNs more biologically plausible [32,33]. Moreover, the use of discrete spike trains instead of analog signals, lower energy requirements due to asynchronous operation, and parallel processing capability make SNNs more suitable for hardware applications than conventional neural networks [34]. The aforementioned advantages of SNNs, together with their fast adaptability, have led to an increasing number of applications of SNNs in various fields, including pattern recognition [35–37], object detection [38–40], and control [41].

SNNs have a similar structure to ANNs, as both consist of layers of neurons connected in a feedforward manner, and there is no feedback among the connections. However, the key difference between these two models is related to data representation. In ANNs, the information that is transmitted along the network depends on the activation value of the neuron and the weight of the connection. In these networks, temporal information is not involved in the transmission or generation of the neuron activity. On the contrary, SNNs transmit the data as discrete, asynchronous spikes and their timing is involved. In ANNs, the output is computed solely from the current input values. On the other hand, in SNNs, previously received spikes have an effect on the current membrane potential, which is biologically more accurate.

Despite the fact that SNNs might be viable alternatives to ANNs, they have a higher computational cost due to the presence of multiple synaptic connections between neurons, each with different weight and delay characteristics. Furthermore, the discrete nature of the spikes makes the development of gradient-based learning rules more difficult. Therefore, the development of new learning algorithms is still an open research topic for spiking neural networks.

The learning methods proposed for SNNs can be classified into three categories, which are supervised learning, unsupervised learning, and reinforcement learning. In this study, the supervised learning of SNNs has been considered. SpikeProp [42] is one of the most widely used supervised learning algorithms for SNNs. It can be considered as a modified version of the backpropagation algorithm of traditional ANNs. In this method error gradients have been determined, and the weights are updated in the opposite direction to minimize the error. A number of improvements and extensions have been proposed for SpikeProp to speed up the training process. In [43], resilient propagation (RProp) and QuickProp algorithms have been adapted to SNNs. In RProp, the learning rate is modified when considering the sign of the gradient, whereas in QuickProp Newton's method is utilized to minimize the error gradient. SuperSpike [44] has been proposed as a further improvement to SpikeProp, in which the derivative of the membrane potential has been considered instead of the firing times. Despite the reported successful results, traditional

methods relying on backpropagation suffer from the possibility of converging to local minima. Moreover, the robustness of these algorithms cannot be guaranteed.

In this study, a novel control scheme utilizing SNNs has been proposed for the trajectory-tracking problem of the quadrotors. Unlike traditional approaches, the update rules governing the behavior of the SNN-based controller have been derived using the Lyapunov stability theorem [45], ensuring the stability of the closed-loop system and guaranteeing convergence to the desired trajectory. To validate the efficacy of the proposed approach, experiments have been conducted in addition to simulation studies, and the results have been compared to the responses of conventional neural networks and PID controllers.

The main advantages of the proposed control strategy can be stated as follows:

- Unlike the traditional neural networks relying on continuous activation functions, the proposed scheme is capable of representing and manipulating temporal signals commonly found in various control systems.
- In the derivation of the parameter update rules, the Lyapunov stability theorem has been exploited to guarantee stable online learning and convergence to the desired trajectory.
- The algorithm provides highly accurate results and has acceptable execution times for real-time applications. A simple network structure with a small number of neurons has been proposed to reduce computational burden and allow deployment in real-time applications.

The remainder of the paper is organized as follows: Section 2 introduces the kinematic and dynamic model of the quadrotor system used in this study. Section 3 describes the structure of the SNN controller and presents the derivation of the update rules using the Lyapunov stability theorem. In Section 4, the results of simulations for different trajectories are given. Section 5 presents the responses obtained from experimental studies. The last section is devoted to concluding remarks.

2. Mathematical Model of the Quadrotor

In this study, a four-rotor rotary wing-type UAV has been used. This type of UAV generally consists of three main parts: airframe, electronic systems, and rotors. The airframe consists of two bars connected at the center and made of as light and durable material as possible to save energy. Rotors are located symmetrically at both ends of the two bars. In this configuration, adjacent propellers rotate in opposite directions, resulting in zero net torque around the central axis, thus preventing uncontrolled rotation of the UAV. The energy applied to each motor is varied according to the trajectory and desired speed. The electronic components are needed to receive the data collected from various sensors (pressure gauges, gyroscopes, compasses, GPS, ultrasonic, etc.) and transmit the signals to the propulsion elements to achieve the desired outputs, such as altitude and attitude.

Three rotational and three translational axes of quadrotors have been illustrated in Figure 1. The terms roll (ϕ) and pitch (θ) are used to describe rotational motions along the longitudinal and lateral axes, respectively, while the yaw angle (ψ) corresponds to rotational motion along the vertical axis of the quadrotor. Moreover, in rotary wing UAVs, the pitch angle is used to produce a movement along the longitudinal axis, while a motion along the lateral axis is provided by adjusting the roll angle.

To determine the altitude and attitude of the rotary wing UAV with six degrees of freedom, two coordinate systems are utilized. The body coordinate system $B = \{X_B, Y_B, Z_B\}$ is attached to the center of the UAV, and the inertial coordinate systems $E = \{X_E, Y_E, Z_E\}$ is defined as fixed relative to a point on the ground. Using the transformation matrix R given in Equation (1), the position and linear velocities given with respect to the coordinate system B can be described by inertial coordinate system E as follows.

$$R = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \quad (1)$$

where $c(\cdot) = cos(\cdot)$ and $s(\cdot) = sin(\cdot)$.

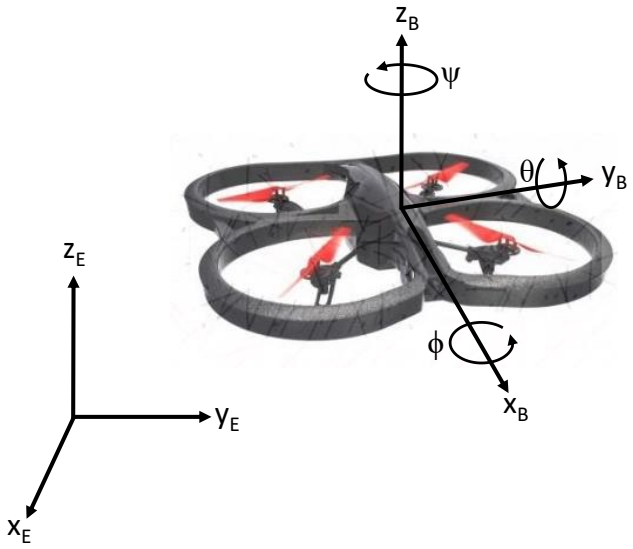


Figure 1. The defined coordinate systems and Euler angles for the quadrotor. The coordinate frame B is attached to the body of the quadrotor and inertial coordinate system matches the initial position of the quadrotor before takeoff.

Using Newton's and Euler's laws of motion, the dynamic model of the system for the rotary wing UAV can be obtained as in Equation (2):

$$\begin{aligned}
\ddot{x} &= \frac{1}{m}((c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi))F_z + D_x) \\
\ddot{y} &= \frac{1}{m}\big((c(\phi)s(\theta)s(\psi) - s(\phi)s(\psi))F_z + D_y\big) \\
\ddot{z} &= \frac{1}{m}((c(\phi)c(\theta))F_z + D_z) - g \\
\ddot{\phi} &= \frac{(I_{yy} - I_{zz})\dot{\theta}\dot{\psi} - J\dot{\theta}(-\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4) + lK_t(\Omega_4^2 - \Omega_2^2)}{I_{xx}} \\
\ddot{\theta} &= \frac{(I_{zz} - I_{xx})\dot{\phi}\dot{\psi} - J\dot{\phi}(-\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4) + lK_t(\Omega_3^2 - \Omega_1^2)}{I_{yy}} \\
\ddot{\psi} &= \frac{(I_{xx} - I_{yy})\dot{\phi}\dot{\theta} + K_d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)}{I_{zz}}
\end{aligned} \tag{2}$$

In these equations, m corresponds to the mass of the drone. I_{xx}, I_{yy}, and I_{zz} are the moment of inertia of the UAV in the x, y, and z directions, respectively. F_z term represents the vertical thrust along z-axis. The drag forces in the x, y, and z directions correspond to the terms D_x, D_y, and D_z. J stands for the moment of inertia of the rotor. Ω_i $(i = 1, 2, 3, 4)$ are the angular speeds of the rotors, whereas l denotes the bar length of the UAV. K_d and K_t correspond to the drag and thrust coefficients, respectively.

3. Controller Design Based on Spiking Neural Networks

Different neuron models have been developed to model spiking neurons, which can be classified according to their similarity to biological neurons and speed. The Spike Response Model (SRM) is among the most widely used models in the literature [46]. This model describes how the incoming spike train generates the new spike train that leaves the neuron. There are different mathematical formulations for the spike response function ε, but they all show a short rise followed by a long period of decay. The spike response function

considered for use in this study is given in Equation (3). The causality has been ensured by setting $\varepsilon(t)$ to zero for negative time values.

$$\varepsilon(t) = \begin{cases} \frac{t}{\tau} e^{1-\frac{t}{\tau}} & t > 0 \\ 0 & t \leq 0 \end{cases} \tag{3}$$

Here, τ controls the steepness of the rise and fall of the function and, more importantly, the location of the peak of the function. In the proposed work, to alleviate the need for a refractoriness term used to prevent an immediate second spike, it is assumed that each neuron is capable of generating a single spike when its potential exceeds the threshold value θ for the first time. In SRM, it is argued that each synaptic connection possesses some delay in transmitting the spikes to the other neurons. Thus, a spike generated at time t_i by neuron i will be received by neuron j at time $t_i + d^k$, where d^k is the amount of latency associated with the synapse k. At time t, the membrane potential of neuron j can be computed using Equation (4):

$$x_j(t) = \sum_{i=1}^{N} \sum_{k=1}^{K} w_{ij}^k \varepsilon(t - t_i - d^k) = \sum_{i=1}^{N} \sum_{k=1}^{K} w_{ij}^k y_i^k(t) \tag{4}$$

where N is the number of presynaptic neurons, and K stands for the number of synapses. w_{ij}^k represents the weight values for the synaptic connection k existing between neuron i and neuron j. When the membrane potential of the postsynaptic neuron j rises above the threshold value at t_j, it will fire a spike and t_j will be recorded as the output of the neuron.

Using spiking neuron models, different network structures can be generated. The most typical form is a feed-forward neural network constructed by combining layers of spiking neurons in a sequential manner. As shown in Figure 2, the inputs and outputs of the network are spike trains. Unlike the second-generation neural networks, multiple synaptic connections exist between presynaptic and postsynaptic neurons, and a different weight value and delay time is assigned to each synaptic connection.

In the literature, supervised learning of spiking neural networks has been generally realized using gradient-based methods [42,47]. In these algorithms, first, a cost function is generated using the difference between the output of the neural network and the desired output. Next, the gradients of this cost function with regard to the network parameters are determined, and the network parameters are updated in the opposite direction of the calculated gradients. Despite the successful results reported throughout the literature, gradient-based learning methods can lead to sub-optimal performance in terms of robustness and convergence speed. Furthermore, convergence is not guaranteed in these learning algorithms as the error surface may contain local minima along with a global minimum, and the system trajectories can become stuck in a local minimum. Among the methods proposed in the literature to mitigate these issues, the approach in which sliding mode control (SMC) theory has been utilized for the adaptation of network parameters has been adopted in this study as it provides stable learning processes and robust system response.

SMC has emerged as a subclass of variable structure systems [48], and it has attracted the interest of many researchers due to its robustness against factors such as disturbances, parameter variations, and uncertainties, which can negatively impact control system performance. This method confines the system trajectories' motion to the sliding surface, a plane where a predetermined error function is zero.

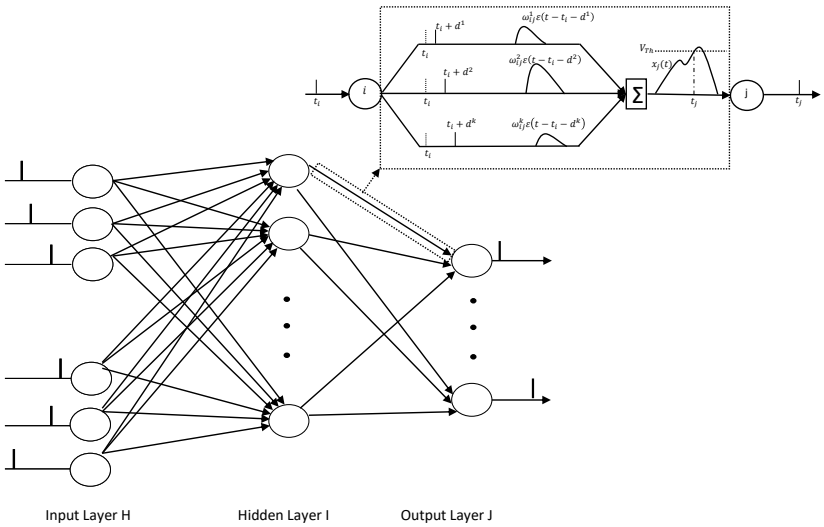


Figure 2. Structure of a fully connected spiking neural network with a single hidden layer. The inputs and outputs of an SNN consist of temporal information, which corresponds to the firing times of the spikes. Neurons in a layer are connected to the neurons of the proceeding layer through multiple synapses. Each synapse has a delay and weight. The network learns by adapting the weight values. In the upper right corner of the figure, a detailed view of synaptic connections has been given. In this figure, the presynaptic neurons are connected to the postsynaptic neurons with k synaptic connections. The spikes generated by neuron i are transmitted to neuron j by these synaptic connections, each having a different delay and weight. At the moment when the membrane potential of the neuron exceeds the threshold value V_{Th} due to the incoming pulses, a spike will be generated and propagated to the post-synaptic neurons.

A sliding surface is generally constructed using the difference between the actual and target outputs of the system and the derivative of this difference. Such a formulation is presented in Equation (5):

$$S(x(t)) = \left(\frac{d}{dt} + \lambda\right)^{(n-1)} (y - y^d) = 0 \tag{5}$$

In this expression, λ is a positive constant denoting the slope of the sliding surface, and y and y^d stand for the system's actual and target outputs, respectively. n indicates the order of differentiation.

For a plane to be considered as a sliding surface, it should meet the two requirements listed below:

- If a trajectory starts or reaches the sliding surface, it should remain and move along this surface for all subsequent times. To satisfy this condition, if $s(x(t)) = 0$, then the derivative of this function must also be zero, i.e., $\dot{s}(x(t)) = 0$.
- If a trajectory starts outside the sliding surface, it should converge to the sliding surface, which requires:

$$\lim_{S \to 0^+} \dot{S} < 0 \quad \text{and} \quad \lim_{S \to 0^-} \dot{S} > 0 \tag{6}$$

In order to determine a sliding surface that satisfies these properties, Lyapunov's stability theorem can be used. A Lyapunov function candidate has been given in (7):

$$V(x,t) = \frac{1}{2} S(x(t))^2 \geq 0 \tag{7}$$

According to the Lyapunov stability theorem, in order to guarantee the stability of the system, the derivative of the given candidate function must be equal to or less than zero.

$$\dot{V}(x,t,S) = S(x(t))\frac{\partial S(x(t))}{\partial t} \leq 0 \tag{8}$$

Therefore, if a function that satisfies the conditions in Equations (7) and (8) can be determined, it can be guaranteed that the trajectories of the system will converge to the sliding surface and "slide" along this surface.

Consider an SNN network with a single hidden layer as shown in Figure 3. In this network structure, the input layer includes H neurons, I neurons exist in the hidden layer, and the output layer consists of J neurons. Moreover, $w_{hi}^{k}(t)$ denotes the synaptic weight between neuron h of the input layer and neuron i of the hidden layer connected by synapse k, while $w_{ij}^{k}(t)$ corresponds to the weight value between neuron j of the output layer and neuron i of the hidden layer for the k-th synapse.

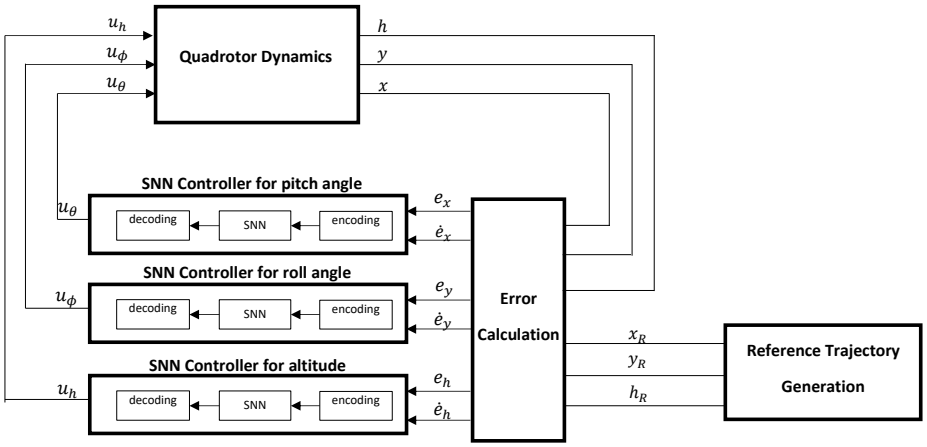


Figure 3. The controller structure. The proposed control scheme utilizes three controllers to attain the trajectory tracking of the quadrotor. Each controller is connected to two inputs, deviation from the reference trajectory along j-axis, e_j, $(j = x, y, h)$, and the rate of change in this deviation, $\dot{e}_j$. These inputs are first converted into spike trains using the encoding scheme given in Equation (21), and applied to the SNN network. The network parameters have been updated using Equations (18) and (19). The output of the network has been decoded into a real value using Equation (22) and applied to the quadrotor system as a control command.

According to the SMC approach, a time-varying sliding surface can be defined as in Equation (9):

$$S_j(e_j(t)) = e_j(t) = \left(t_j^d(t) - t_j(t)\right) = 0 \tag{9}$$

In this expression, the difference between the actual firing time, $t_j(t)$, and its target value, $t_j^d(t)$, has been employed in the computation of the learning error of output neuron j.

In this study, Lyapunov function candidates have been defined using the squared value of learning errors as presented in Equation (10):

$$V_j(S_j(t)) = \frac{1}{2}S_j^2(t) = \frac{1}{2}e_j^2(t) = \frac{1}{2}\left(t_j^d(t) - t_j(t)\right)^2 \tag{10}$$

The reachability condition requires $V_j(S_j(t))\dot{V}_j(S_j(t)) \leq 0$. Hence, the derivative of the Lyapunov function candidate should satisfy the following condition:

$$\dot{V}_j(S_j(t)) = S_j(t)\dot{S}_j(t) = e_j(t)\dot{e}_j(t) = e_j(\dot{t}_j^d - \dot{t}_j) \leq 0 \tag{11}$$

Without loss of generality, it is assumed that the desired firing time of the output neuron is constant, i.e., $\dot{t}_j^d = 0$. The derivative of the Lyapunov function candidate requires the computation of the derivative of the output neuron's firing time with respect to time, $\dot{t}_j$. For this purpose, the chain rule has been applied as given in Equation (12).

$$\frac{dt_j}{dt} = \frac{\partial t_j}{\partial w_{ij}^k}\frac{dw_{ij}^k}{dt} + \frac{\partial t_j}{\partial w_{hi}^k}\frac{dw_{hi}^k}{dt} = \frac{\partial t_j}{\partial w_{ij}^k}\dot{w}_{ij}^k + \left(\sum_{j=1}^{J}\frac{\partial t_j}{\partial t_i}\right)\frac{\partial t_i}{\partial w_{hi}^k}\dot{w}_{hi}^k \tag{12}$$

To determine the first term of Equation (12), $\partial t_j/\partial w_{ij}^k$, the chain rule can be used as noted in Equation (13):

$$\frac{\partial t_j}{\partial w_{ij}^k} = \frac{\partial t_j}{\partial x_j(t_j)}\frac{\partial x_j(t_j)}{\partial w_{ij}^k} = \frac{-y_i^k(t_j)}{\sum_{i=1}^{I}\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}} \tag{13}$$

The next partial derivative of Equation (12) corresponds to the derivative of the output neurons' firing times with respect to the firing times of the neurons in the hidden layer, which can be computed as:

$$\frac{\partial t_j}{\partial t_i} = \frac{\partial t_j}{\partial x_j(t_j)}\frac{\partial x_j(t_j)}{\partial t_i} = \frac{-\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_i}}{\sum_{i=1}^{I}\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}} \tag{14}$$

The last term of Equation (12) refers to the partial derivative of the firing times of hidden layer neurons with respect to the weights can be computed as:

$$\frac{\partial t_i}{\partial w_{hi}^k} = \frac{\partial t_i}{\partial x_i(t_i)}\frac{\partial x_i(t_i)}{\partial w_{hi}^k} = \frac{-y_h^k(t_i)}{\sum_{h=1}^{H}\sum_{k=1}^{K} w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}} \tag{15}$$

Using the partial derivatives derived in Equations (13)–(15), the derivative of the output neurons' firing times with respect to time, $\dot{t}_j$, can be rewritten as given in Equation (16):

$$\frac{dt_j}{dt} = \frac{-y_i^k(t_j)}{\sum_{i=1}^{I}\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}}\dot{w}_{ij}^k + \left(\sum_{j}^{J}\frac{\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_i}}{\sum_{i=1}^{I}\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}}\right)\frac{y_h^k(t_i)}{\sum_{h=1}^{H}\sum_{k=1}^{K} w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}}\dot{w}_{hi}^k \tag{16}$$

Using the expression of $\dot{t}_j$ given in Equation (16), the derivative of the Lyapunov function candidate, $\dot{V} = -e_j\dot{t}_j$ results in:

$$\dot{V}_j(S_j(t)) = \frac{y_i^k(t_j)e_j\dot{w}_{ij}^k}{\sum_{i=1}^{I}\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}} - \frac{\left(\sum_{j}^{J} e_j \sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_i}\right) y_h^k(t_i)\dot{w}_{hi}^k}{\left(\sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}\right)\left(\sum_{h=1}^{H}\sum_{k=1}^{K} w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}\right)} \tag{17}$$

In this study, the update rules are selected as follows:

$$\dot{w}_{ij}^k = -\alpha \sum_{i=1}^{I}\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j} sgn(e_j) \tag{18}$$

$$\dot{w}_{hi}^k = \beta sgn\left(\sum_{j=1}^{J} e_j \frac{\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_i}}{\sum_{i=1}^{I}\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}}\right) \sum_{h=1}^{H}\sum_{k=1}^{K} w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i} \tag{19}$$

Using the weight update rules given in Equations (18) and (19) in Equation (17) to replace $\dot{w}_{hi}^k$ and $\dot{w}_{hi}^k$ results in:

$$\dot{V}_j(S_j(t)) = -\alpha y_i^k(t_j)|e_j| - \beta sgn\left|\sum_{j=1}^{J} e_j \frac{\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_i}}{\sum_{i=1}^{I}\sum_{k=1}^{K} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}}\right| y_h^k(t_i) \tag{20}$$

which implies that $\dot{V}_j(S_j(t)) \leq 0$ for the selected weight update rules, and stable online learning can be achieved as $V_j(S_j(t))\dot{V}_j(S_j(t)) \leq 0$ has been attained for all times.

In this study, a separate controller has been designed for each axis, with each controller having two inputs and a single output. The input signals for the roll angle controller have been selected as the deviation of the quadrotor from the target trajectory along the y-axis, e_y, and the rate of change in this deviation is denoted by $\dot{e}_y$. In a similar manner, the difference, e_x, between the actual and target positions along the x-axis and the time derivative of this value, $\dot{e}_x$, have been fed as input signals to the controller regulating the pitch angle. In the controller used to determine the height of the UAV from the ground, the difference between the actual and desired altitudes, e_h, and the derivative of this difference, $\dot{e}_h$, have been used.

In the input layer of the proposed controller, first, the inputs have been converted into spike times using a latency coding scheme, in which higher signal values correspond to an earlier firing time than the smaller input signals. This scheme relies on the idea that a higher stimulus is more likely to generate a larger increase in the membrane potential resulting in the firing of the neuron earlier. To convert the controller inputs into spike times, the following equation has been utilized:

$$t_j(x) = t_{max} - \left(t_{min} + \frac{(x - x_{min})(t_{max} - t_{min})}{x_{max} - x_{min}}\right) \tag{21}$$

where $t_j(x)$ is the firing time of neuron j with t_{min} and t_{max} being the minimum and the maximum spike times allowed. x_{max} and x_{min} are the minimum and maximum values of the input signals and x corresponds to the current value of the input variable.

The encoded input signals are fed to the SNN, and they are propagated through the network to generate the output spike train, while computing the output spikes, the synaptic weights of the controller have been updated using Equations (19) and (18).

The outputs of SNNs are spike trains. To be able to use these temporal data in control systems, they should be converted into real numbers. The following formula has been used for this purpose:

$$u = u_{min} + \frac{(t_{max} - t_j)(u_{max} - u_{min})}{(t_{max} - t_{min})} \tag{22}$$

where u_{min} and u_{max} are the minimum and maximum values of the control signals that will be applied to the plant. t_j is the computed spike time of the output neuron j, whereas t_{min} and t_{max} denote the minimum and the maximum allowable spike times.

4. Simulated Studies

The simulations and experimental studies have been carried out on a Parrot AR.Drone 2.0 (French company Parrot, Paris, France) model UAV because of its low cost and easy availability. This UAV has been widely used in many research projects as it allows the users to implement their own controller designs by using a variety of programs, including Matlab (R2016a or later), Python (version 2.6.6 or later), and Labview (version 9.0 or later) [49,50]. The parameters for the quadrotor used in this study are given in Table 1 [51–53].

The communication between the controlling computer and Ar.Drone 2.0 can be established via an IEEE 802.11 wireless connection. Two sets of data can be acquired from the UAV. The first set is related to the navigation, which includes pressure, altitude, speed, and angle values obtained from the sensors. The next set of data is a video stream obtained using either front or bottom camera.

Table 1. Parameters of Parrot AR.Drone 2.0 UAV.

Parameter	Value
m	0.445 kg
l	0.125 m
I_{xx}	2.7×10^{-3} kg·m^2
I_{yy}	2.9×10^{-3} kg·m^2
I_{zz}	5.3×10^{-3} kg·m^2
J	2.03×10^{-5} kg·m^2
K_d	2.38×10^{-9} N·s^2
K_t	9.14×10^{-6} N·m·s^2

From the data acquired from the sensors, the following can be used in the design of various controllers:

z: altitude, [m];
v_x: linear velocity along x-axis, [m/s];
v_y: linear velocity along y-axis, [m/s];
ϕ: roll, [rad];
θ: pitch, [rad];
ψ: yaw, [rad].

The following motion control commands scaled to the range $[-1, 1]$ can be used to direct the movements of the quadrotor:

u_z: linear velocity command in z direction;
$u_{\dot{\psi}}$: angular velocity about the yaw axis;
u_ϕ: roll command;
u_θ: pitch command.

The roll command is used to create a leftward/rightward tilt angle. Negative values cause the UAV to move along the $(-)y$ direction by tilting to the left. The pitch angle command is used to create a forward/backward tilt angle. Negative values cause the UAV to move in the $(+)x$ direction by tilting forward.

Simulation studies have been carried out in MATLAB/Simulink [54] environment to validate the proposed learning algorithm's effectiveness. Three different target trajectories have been used in the simulation studies. The first of these trajectories, target trajectory 1, has been defined in the xy-plane in two dimensions and requires abrupt direction changes. The other target trajectories require the simultaneous generation of control signals in all three axes. The target trajectories are shown in Figure 4.

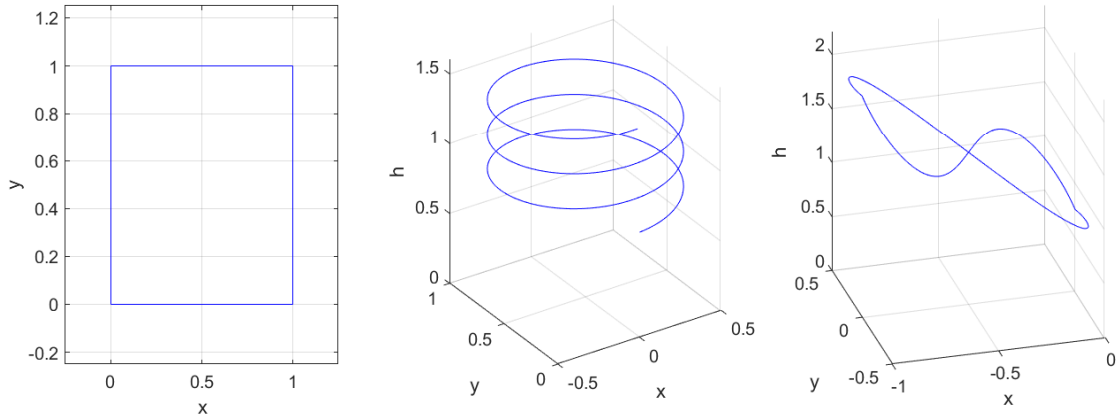


Figure 4. Target trajectories used in simulations and experiments. Three different reference trajectories have been generated. The first reference trajectory has been defined on xy-plane and it requires sudden changes in the movements of the quadrotor. The other reference trajectories require simultaneous control of the quadrotor movement along each translational axis.

For the encoding step, where sensor inputs are converted into the spike times, t_{max} and t_{min} have been set to 18 ms and 2 ms, respectively. The error signal is assumed to have a value within the range $[-1, 1]$, whereas the derivative of error is assumed to vary between -0.2 and 0.2. The sampling time has been set to 20 ms.

The presence of multiple synapses between interconnected neurons has an adverse effect on the computational burden of networks consisting of spiking neurons. To lessen this load, the number of hidden layer neurons and synapses should be optimized to reduce the computation time and yet provide accurate results. In this study, the number of neurons in the hidden layer has been set to 10 while the number of synapses has been selected as 4 to provide high accuracy with acceptable execution time.

In the decoding of spike times into the real values, t_{max} and t_{min} are assumed to have values of 20 ms and 12 ms, respectively. The computed control command u_j, $(j = \phi, \theta, \psi)$, passes through saturation operation, as the quadrotor used in this study requires control commands to be within the range $[-1, 1]$.

To validate the performance of the designed controller, simulations have been repeated for PID and MLP controllers. The parameters of the PID controller have been determined using the PID tuner feature of Matlab/Simulink. The values obtained from this block have been empirically tuned to provide the best response in simulation and experimental studies after considering the possible differences between the real system and the mathematical model of the system (differences due to simplification and assumptions, parameter uncertainties, etc.). In the literature, it is seen that empirical approaches are generally adopted for PID controllers used in the control of rotary wing UAVs [55]. The coefficients of PID controllers were determined as $K_P = 0.6$, $K_I = 0.01$, and $K_D = 0.02$ for the x and y axes and $K_P = 2$, $K_I = 0.01$, and $K_D = 0.35$ for the z axis. For the MLP, which is the predominant structure of the second generation of networks, 20 neurons have been used in the hidden layer, as a smaller number of neurons could not provide successful results in tracking. In ANNs, the performance of the controller depends on the selection of activation functions. To select the best-performing MLP structure, simulation studies have been carried out for the MLP networks with sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU) activation functions. The average RMSE values and standard deviation have been provided in Table 2. The ReLU activation function has been used in the rest of simulation studies, as it provides better performance in terms of RMSE values for most trajectories.

Table 2. Average RMSE values and standard deviation (STD) obtained for different activation functions in the absence of disturbance. STD values are given under the average RMSE values inside parentheses. (All units are in meters).

Activation Function	Trajectory 1		Trajectory 2			Trajectory 3		
	X_{RMSE} (STD)	Y_{RMSE} (STD)	X_{RMSE} (STD)	Y_{RMSE} (STD)	Z_{RMSE} (STD)	X_{RMSE} (STD)	Y_{RMSE} (STD)	Z_{RMSE} (STD)
sigmoid	0.222 (0.012)	0.318 (0.006)	0.047 (0.009)	0.093 (0.010)	0.024 (0.008)	0.051 (0.012)	0.073 (0.009)	0.075 (0.012)
tanh	0.201 (0.012)	0.245 (0.010)	0.070 (0.005)	0.087 (0.008)	0.028 (0.004)	0.034 (0.013)	0.088 (0.007)	0.031 (0.014)
ReLU	0.155 (0.009)	0.186 (0.009)	0.079 (0.005)	0.062 (0.011)	0.017 (0.005)	0.045 (0.010)	0.051 (0.011)	0.061 (0.010)

The system responses for target trajectory 1, target trajectory 2, and target trajectory 3 are shown in Figure 5. The results illustrate that all controllers are capable of tracking the target trajectories accurately in the absence of disturbing factors in the operating environment. The root mean square error (RMSE) method has been employed to evaluate the developed controllers' performance. For each trajectory and for each controller, the simulations have been run 10 times. Average RMSE values and standard deviation of the metric have been presented in Table 3. According to this table, the largest error values for

the SNN controller are obtained for target trajectory 1, which requires sudden maneuvers. The reason for larger RMSE values might be devoted to the time required to adapt the parameters in case of sudden reference changes.

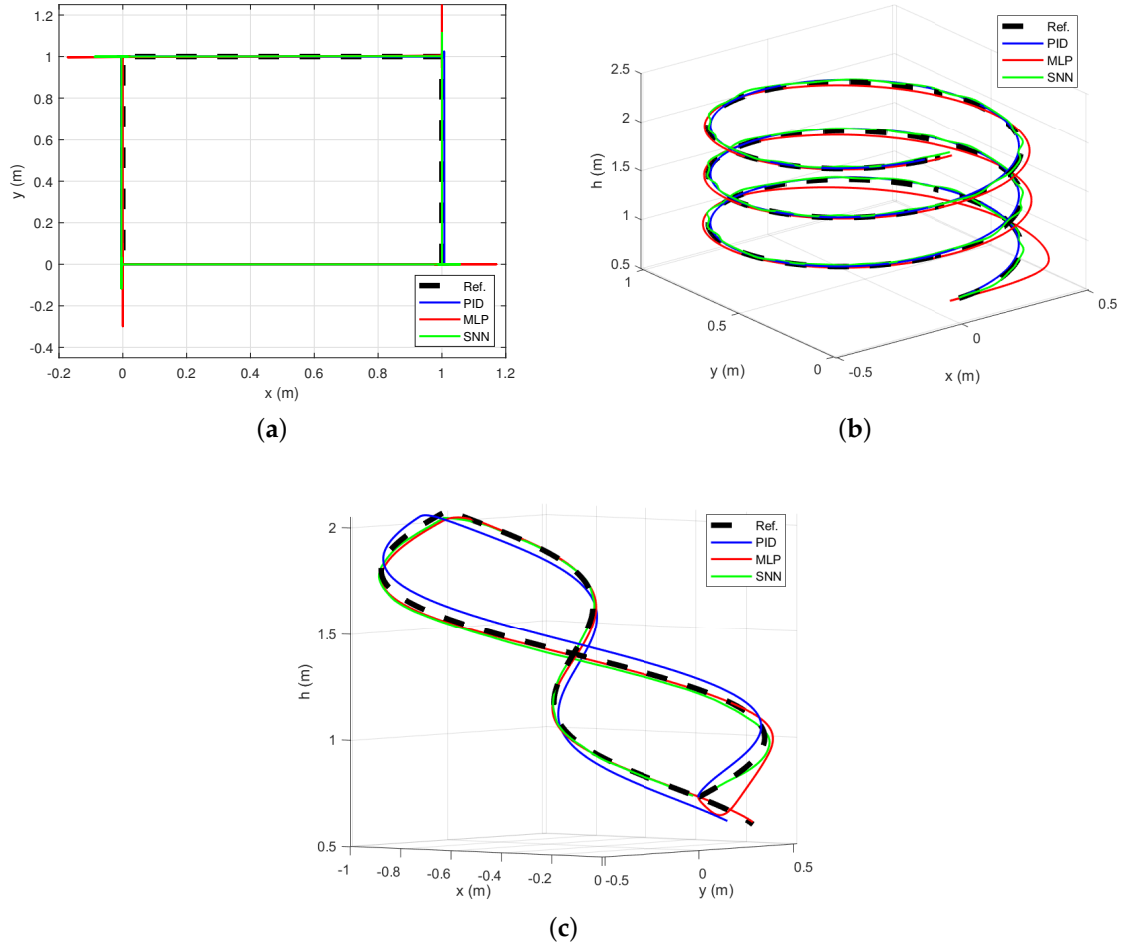


Figure 5. Simulation results in the absence of disturbing factors for (**a**) target trajectory 1, (**b**) target trajectory 2, and (**c**) target trajectory 3.

Table 3. Average RMSE values and standard deviation (STD) obtained for PID, MLP, and SNN controllers in the absence of disturbance. For MLP and SNN controllers, STD values are given under the average RMSE values inside parentheses. (All units are in meters).

Controller	Trajectory 1		Trajectory 2			Trajectory 3		
	X_{RMSE} (STD)	Y_{RMSE} (STD)	X_{RMSE} (STD)	Y_{RMSE} (STD)	Z_{RMSE} (STD)	X_{RMSE} (STD)	Y_{RMSE} (STD)	Z_{RMSE} (STD)
PID	0.116	0.116	0.082	0.070	0.041	0.058	0.136	0.028
MLP	0.155 (0.009)	0.186 (0.009)	0.070 (0.001)	0.087 (0.002)	0.028 (0.001)	0.045 (0.010)	0.051 (0.011)	0.061 (0.010)
SNN	0.154 (0.010)	0.175 (0.010)	0.043 (0.004)	0.031 (0.002)	0.016 (0.004)	0.032 (0.009)	0.049 (0.010)	0.059 (0.006)

To verify the performance of the controllers under external disturbances such as wind, a disturbance of $d(t) = 0.2sin(2.5\pi t)$ has been added to the simulation in x- and y-directions. The responses of the UAV for the same target trajectories in the presence of disturbance have been shown in Figure 6, and the trajectory tracking performances of the controllers for x- and y-axes are shown in Figures 7 and 8. It can be observed from these

figures that the spiking neural network controller is able to track the target trajectory with less deviation when disturbances affect the system. The RMSE values obtained for the simulation study are given in Table 4. Upon analysis of these data, it is evident that the proposed controller yields smaller error values than the other controllers utilized in this work and that the impact of disturbance on controller performance is more limited.

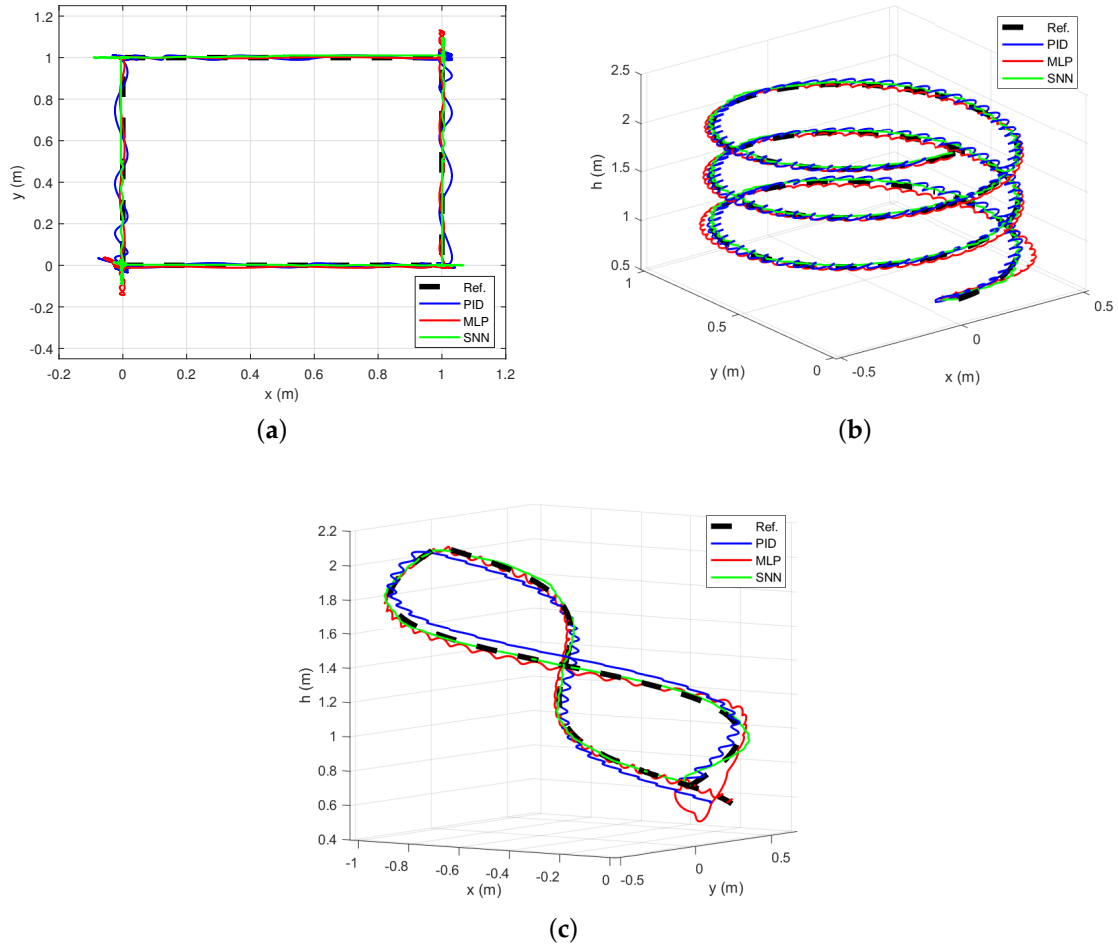
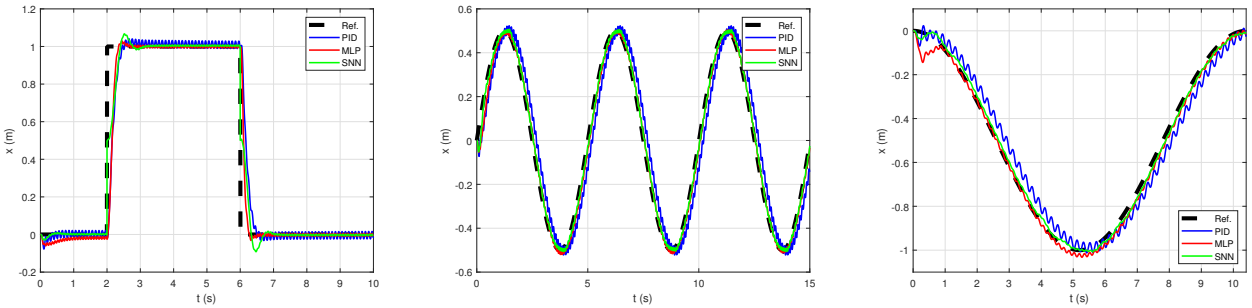


Figure 6. Simulation results in the presence of sinusoidal disturbing factors for (**a**) target trajectory 1, (**b**) target trajectory 2, and (**c**) target trajectory 3. To imitate the effect of a wind disturbance acting along x- and y-directions, a disturbance of $d(t) = 0.2sin(2.5\pi t)$ has been included in x- and y-axis.

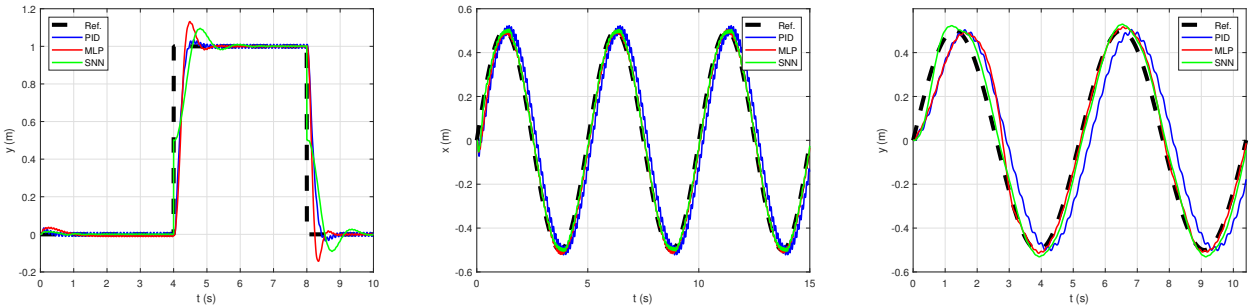
Table 4. Average RMSE values and standard deviation (STD) obtained for PID, MLP, and SNN controllers in the presence of sinusoidal disturbance. For MLP and SNN controllers, STD values have been specified next to the average RMSE values inside parentheses. (All units are in meters).

	Trajectory 1		Trajectory 2		Trajectory 3	
Controller	X_{RMSE} **(STD)**	Y_{RMSE} **(STD)**	X_{RMSE} **(STD)**	Y_{RMSE} **(STD)**	X_{RMSE} **(STD)**	Y_{RMSE} **(STD)**
PID	0.161	0.164	0.094	0.091	0.071	0.143
MLP	0.154 (0.009)	0.178 (0.012)	0.039 (0.007)	0.058 (0.010)	0.055 (0.009)	0.069 (0.012)
SNN	0.145 (0.007)	0.163 (0.008)	0.034 (0.004)	0.021 (0.009)	0.044 (0.008)	0.048 (0.010)



(**a**) (**b**) (**c**)

Figure 7. Responses of the controllers along x-direction in the presence of sinusoidal disturbing factors acting in x- and y- directions: (**a**) target trajectory 1, (**b**) target trajectory 2, and (**c**) target trajectory 3.



(**a**) (**b**) (**c**)

Figure 8. Responses of the controllers along y-direction in the presence of sinusoidal disturbing factors acting in x- and y- directions: (**a**) target trajectory 1, (**b**) target trajectory 2, and (**c**) target trajectory 3.

In the next part of the simulations, to testify the robustness of the controllers, a discrete wind gust model [56], as given in Equation (23), has been used to simulate the effects of wind.

$$V_{wj} = V_{wj1} + \frac{V_{wj2} - V_{wj1}}{2}\left(1 - cos\left(\frac{\pi(t - t_1)}{t_2 - t_1}\right)\right) \tag{23}$$

where $j = x, y, z$. In this formulation, it is assumed that the wind velocity V_{wj} affecting in the j direction has a value of V_{wj1} at t_1 and V_{wj2} at t_2. The generated wind affects the quadrotor as a drag force given as in Equation (24) [57]:

$$d_{wj} = -0.5\rho k_j (V_j - V_{wj})^2 sgn(v_j - v_{wj}) \tag{24}$$

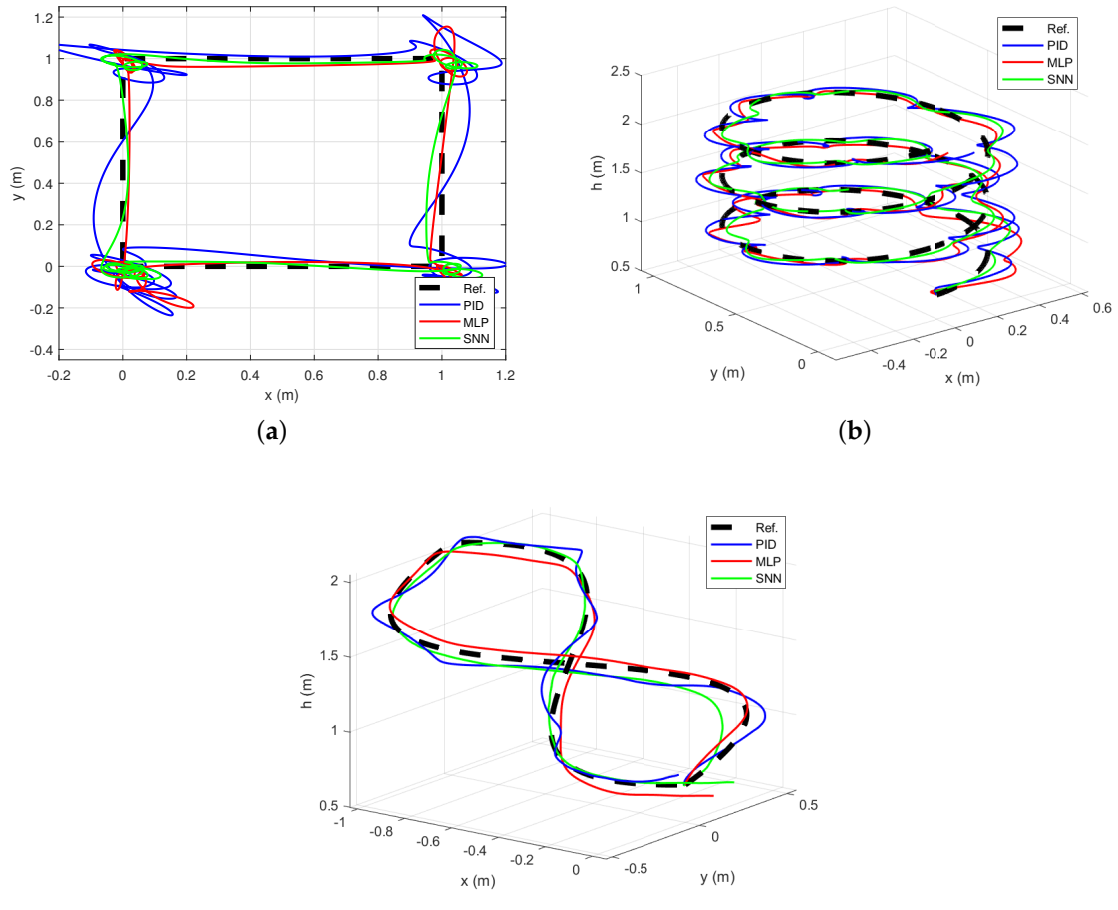
where ρ is the air density, v_j is the velocity of the quadrotor along the inertial axis j, and k_j terms correspond to scaling factors.

It is assumed that along with the wind disturbance, the system is also subjected to fluctuations in the lift force, which are modeled by d_{x1}, d_{y1}, and d_{z1}. The total disturbance affecting the quadrotor system can be written as $d_j = d_{j1} + d_{wj}$ with $(j = x, y, z)$. In the simulations, the parameters given in Table 5 have been used to imitate the effect of wind [57].

Table 5. Parameters used in the discrete wind gust model.

Parameter	Value
ρ	1.225 kg/m^3
$V_{wj1}, (j = x, y, z)$	0 m/s
$V_{wj2}, (j = x, y, z)$	-4 m/s
t_1	20 s
t_2	30 s
d_{x1}	$sin(t)$
d_{y1}	$cos(t)$
d_{x1}	$sin(t)cos(t)$

Generated disturbances have been applied to the quadrotor model in all three directions. The scaling factor k_j given in Equation (24) has been set to 0.025. From the system responses given in Figure 9 it can be observed that all three controllers have been adversely affected by the external disturbance. The response of the PID controller is more oscillatory compared to the other controllers used in this study. The average RMSE values, along with the standard deviations, have been presented in Table 6. The proposed SNN controller provided smaller RMSE values, except the z-direction of the target trajectory 3, for which the PID controller attains the smallest error.



(**a**) (**b**)

(**c**)

Figure 9. Simulation results for discrete wind gust model for (**a**) target trajectory 1, (**b**) target trajectory 2, and (**c**) target trajectory 3. The disturbing factors acting in all three axes have been generated using the discrete wind gust model given in Equation (24).

Table 6. Average RMSE values and standard deviation (STD) obtained for PID, MLP, and SNN controllers in the presence of disturbance generated using discrete wind gust model. For MLP and SNN controllers, STD values are given under the average RMSE values inside parentheses. (All units are in meters).

Controller	Trajectory 1		Trajectory 2			Trajectory 3		
	X_{RMSE} (STD)	Y_{RMSE} (STD)	X_{RMSE} (STD)	Y_{RMSE} (STD)	Z_{RMSE} (STD)	X_{RMSE} (STD)	Y_{RMSE} (STD)	Z_{RMSE} (STD)
PID	0.166	0.165	0.095	0.076	0.028	0.062	0.136	0.028
MLP	0.161 (0.007)	0.171 (0.009)	0.068 (0.004)	0.061 (0.005)	0.033 (0.007)	0.054 (0.015)	0.055 (0.017)	0.066 (0.016)
SNN	0.153 (0.010)	0.152 (0.008)	0.041 (0.007)	0.059 (0.010)	0.014 (0.006)	0.048 (0.011)	0.052 (0.013)	0.054 (0.011)

5. Experimental Results

After the simulation studies, real-time experiments have been carried out for the same trajectories. In the experimental studies, data transfer with the rotary wing UAV has been established by Wi-Fi over an 802.11n access point-based wireless network. The sampling time has been chosen as 0.02 s to ensure a stable communication. Simulink Desktop Real-Time in the MATLAB environment has been used to deploy the proposed controllers. The attitude information of the UAV has been obtained using the velocity values along the longitudinal and lateral axes provided by the sensor board. The altimeter located on the bottom surface of the quadrotor has been used to determine the altitude of the UAV. All experiments have been carried out indoors in a controlled environment.

In the first set of experiments, the square target trajectory 1 has been used to investigate the behavior of the controller in the presence of sudden changes in the reference directions. The experiment has been started after the UAV took off and reached a height of 75 cm. The experimental results for the square target trajectory have been presented in Figure 10a. From these results, it can be observed that the proposed SNN controller provides less deviation from the reference value in both directions.

In the second part of the experiments, target trajectories using sinusoidal functions in the x- and y-directions and linear functions in the z-axis have been used. At the beginning of the experiment, similar to the previous experiment, after the UAV reached an altitude of 75 cm, the reference trajectory has been transmitted to the UAV over the wireless network. When the system responses shown in Figure 10b,c are analyzed, it can be seen that the spiking neural network controller is able to achieve less deviation from the target trajectory. The RMSE values obtained for all three trajectory responses are given in Table 7.

Table 7. RMSE values obtained in real-time experiments for PID, MLP, and SNN controllers. The control loop starts when the quadrotor reaches an altitude of 75 cm. The experiments have been conducted 10 times, and the average RMSE values and standard deviation have been computed. (All units in meters).

Controller	Trajectory 1		Trajectory 2			Trajectory 3		
	X_{RMSE} (STD)	Y_{RMSE} (STD)	X_{RMSE} (STD)	Y_{RMSE} (STD)	Z_{RMSE} (STD)	X_{RMSE} (STD)	Y_{RMSE} (STD)	Z_{RMSE} (STD)
PID	0.322 (0.020)	0.363 (0.019)	0.128 (0.020)	0.110 (0.023)	0.052 (0.020)	0.069 (0.021)	0.145 (0.019)	0.147 (0.016)
MLP	0.315 (0.021)	0.342 (0.024)	0.096 (0.013)	0.078 (0.026)	0.073 (0.023)	0.071 (0.020)	0.107 (0.020)	0.175 (0.024)
SNN	0.301 (0.022)	0.324 (0.021)	0.121 (0.020)	0.095 (0.023)	0.037 (0.016)	0.057 (0.022)	0.096 (0.023)	0.124 (0.018)

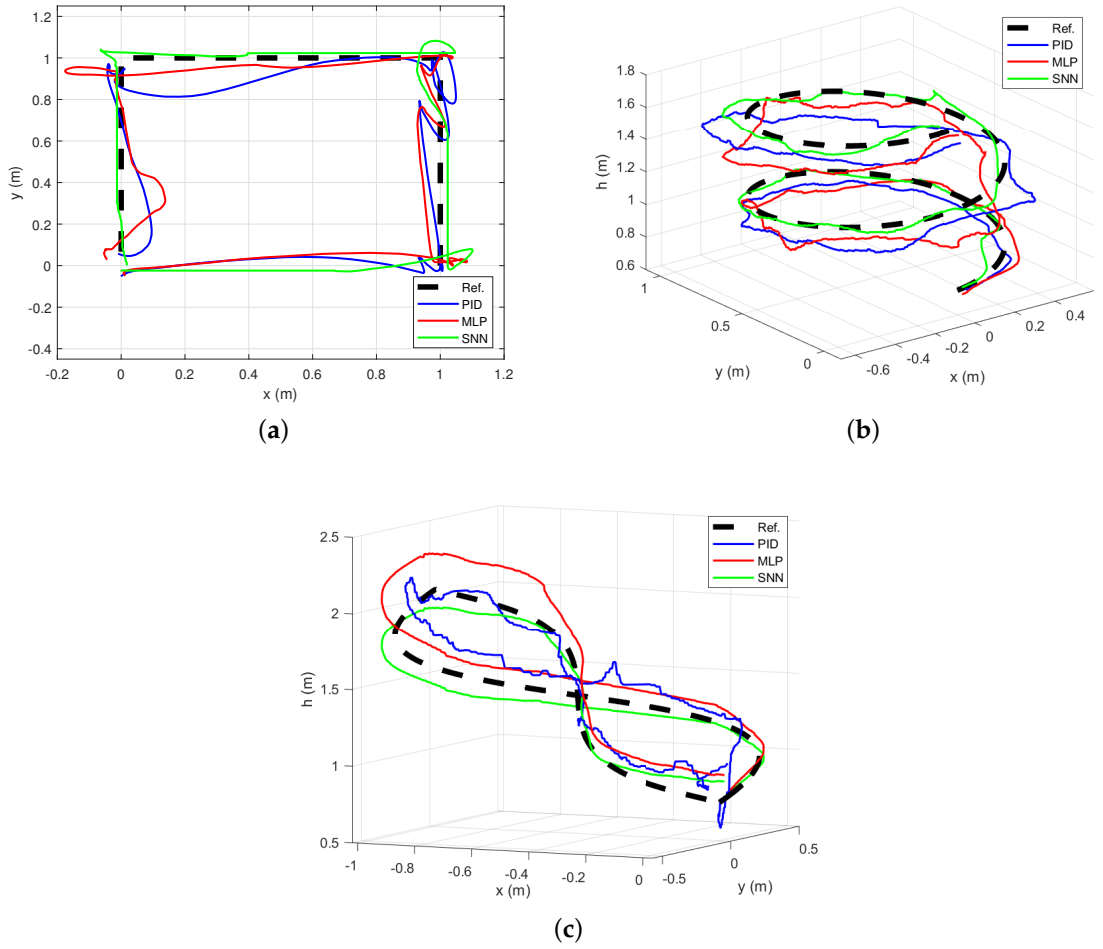


Figure 10. Experimental results for (**a**) target trajectory 1, (**b**) target trajectory 2, and (**c**) target trajectory 3. Reference trajectory and control commands have been transmitted to the quadrotor once the altitude reaches 75 cm.

6. Conclusions

This study intended to investigate the potential of this third generation of networks for the trajectory tracking problem of quadrotors. In the design of the learning algorithm, the sliding mode control theory has been used instead of the conventional gradient descent method to ensure a robust system response and stable online learning. The Lyapunov stability theorem has been utilized in the derivation of parameter update rules. A separate controller has been used to regulate the movements of the quadrotor along each direction. The deviation from the reference trajectory and its derivative have been employed as the controller input signals, which are converted to spike trains using decoding. Online learning of the networks continues throughout the simulations and experiments. The resulting spike times at the output of the network have been converted to real values and applied to the system as control inputs.

Despite the computational load of SNNs, successful results have been obtained for the proposed learning algorithm in real-time experiments and simulation studies. Both simulation and experimental studies have been performed for three different trajectories to validate the effectiveness of the proposed scheme. In order to better evaluate the obtained results, the same studies have been repeated for PID controllers and MLP-based controllers. The results obtained from simulated and experimental studies are in line with the statements which claim that SNNs are capable of providing at least the same performance as their second-generation counterparts [58], as less deviation from the target trajectory has been

attained compared to model-based and second-generation neural network controllers with a robust system response.

Funding: This research was funded by the Bilgi Research Fund of Istanbul Bilgi University (Project no: AK 850890000).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The author declares no conflicts of interest.

Nomenclature/Symbols

The following symbols have been used in this manuscript:

$d_j, (j = x, y, z)$	disturbance along z-direction
d^k	delay for $k-$th synapse
$D_j, (j = x, y, z)$	drag force in j-direction
F_z	thrust along z-axis
$I_{jj}, (j = x, y, z)$	moment of inertia along j-axis
J	moment of inertia of the rotor
m	mass of quadrotor
$\Omega_i, (i = 1, \cdot, 4)$	angular speeds of the rotors
l	bar length of the UAV
K_d	drag coefficient
K_t	thrust coefficient
$S(x(t))$	sliding surface
t_i	firing time of the $i-$th neuron
t_i^d	desired firing time of the $i-$th neuron
t_{max}	maximum allowable time for a neuron to fire
t_{min}	minimum time required for a neuron to fire
u_z	linear velocity command in z direction
$u_{\dot{\psi}}$	angular velocity about the yaw axis
u_ϕ	roll command
u_θ	pitch command
$V(x, t)$	Lyapunov function candidate
$V_j, (j = x, y, z)$	speed along j-direction
$V_{wj}, (j = x, y, z)$	wind velocity along j-direction
w_{ij}^k	synaptic weight of the connection between $i-$th neuron and $j-$th neuron of the proceeding layer for synapse k
x_j	membrane potential of the $j-$th neuron
$\epsilon(t)$	spike response function
θ	pitch angle
τ	time decay constant
ρ	air density
ϕ	roll angle
ψ	yaw angle

References

1. Mishra, B.; Garg, D.; Narang, P.; Mishra, V. Drone-surveillance for search and rescue in natural disaster. *Comput. Commun.* **2020**, *156*, 1–10.
2. Martinez-Alpiste, I.; Golcarenarenji, G.; Wang, Q.; Alcaraz-Calero, J.M. Search and rescue operation using UAVs: A case study. *Expert Syst. Appl.* **2021**, *178*, 114937.
3. Adade, R.; Aibinu, A.M.; Ekumah, B.; Asaana, J. Unmanned Aerial Vehicle (UAV) applications in coastal zone management—A review. *Environ. Monit. Assess.* **2021**, *193*, 1–12.
4. Shahmoradi, J.; Talebi, E.; Roghanchi, P.; Hassanalian, M. A comprehensive review of applications of drone technology in the mining industry. *Drones* **2020**, *4*, 34. [CrossRef]
5. Ollero, A.; Merino, L. Unmanned aerial vehicles as tools for forest-fire fighting. *For. Ecol. Manag.* **2006**, *234*, S263.
6. Yazid, E.; Garratt, M.; Santoso, F. Position control of a quadcopter drone using evolutionary algorithms-based self-tuning for first-order Takagi–Sugeno–Kang fuzzy logic autopilots. *Appl. Soft Comput.* **2019**, *78*, 373–392.

7. Mellinger, D.; Michael, N.; Kumar, V. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *Int. J. Robot. Res.* **2012**, *31*, 664–674.
8. Bouabdallah, S.; Noth, A.; Siegwart, R. PID vs LQ control techniques applied to an indoor micro quadrotor. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, Japan, 28 September–2 October 2004; Volume 3, pp. 2451–2456. [CrossRef]
9. Cowling, I.D.; Yakimenko, O.A.; Whidborne, J.F.; Cooke, A.K. Direct method based control system for an autonomous quadrotor. *J. Intell. Robot. Syst.* **2010**, *60*, 285–316.
10. Alexis, K.; Papachristos, C.; Nikolakopoulos, G.; Tzes, A. Model predictive quadrotor indoor position control. In Proceedings of the 2011 19th Mediterranean Conference on Control & Automation (MED), Corfu, Greece, 20–23 June 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1247–1252.
11. Abdolhosseini, M.; Zhang, Y.M.; Rabbath, C.A. An efficient model predictive control scheme for an unmanned quadrotor helicopter. *J. Intell. Robot. Syst.* **2013**, *70*, 27–38.
12. 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.
13. Meda, J. Estimation of complex systems with parametric uncertainties using a JSSF heuristically adjusted. *IEEE Lat. Am. Trans.* **2018**, *16*, 350–357.
14. Meda-Campana, J.A. On the estimation and control of nonlinear systems with parametric uncertainties and noisy outputs. *IEEE Access* **2018**, *6*, 31968–31973.
15. Mehndiratta, M.; Kayacan, E.; Reyhanoglu, M.; Kayacan, E. Robust tracking control of aerial robots via a simple learning strategy-based feedback linearization. *IEEE Access* **2019**, *8*, 1653–1669.
16. Wang, C.; Fan, H.; Qiang, X. A Review of Uncertainty-Based Multidisciplinary Design Optimization Methods Based on Intelligent Strategies. *Symmetry* **2023**, *15*, 1875. [CrossRef]
17. Tambe, M. Towards flexible teamwork. *J. Artif. Intell. Res.* **1997**, *7*, 83–124.
18. Fraden, J. *Handbook of Modern Sensors*; Springer: Berlin/Heidelberg, Germany, 1999.
19. Bordoni, F.; D'Amico, A. Noise in sensors. *Sens. Actuators Phys.* **1990**, *21*, 17–24.
20. Nakasone Nakamurakari, S.M. Reinforcement Learning for an Attitude Control Algorithm for Racing Quadcopters. Master's Thesis, Tecnologico de Monterrey, Monterrey, Mexico, 2022.
21. Abichandani, P.; Lobo, D.; Ford, G.; Bucci, D.; Kam, M. Wind measurement and simulation techniques in multi-rotor small unmanned aerial vehicles. *IEEE Access* **2020**, *8*, 54910–54927.
22. Guo, H.; Hong, H. Research on filtering algorithm of MEMS gyroscope based on information fusion. *Sensors* **2019**, *19*, 3552. [CrossRef]
23. Alfian, R.I.; Ma'arif, A.; Sunardi, S. Noise reduction in the accelerometer and gyroscope sensor with the Kalman filter algorithm. *J. Robot. Control (JRC)* **2021**, *2*, 180–189.
24. Mo, H.; Farid, G. Nonlinear and adaptive intelligent control techniques for quadrotor uav—A survey. *Asian J. Control* **2019**, *21*, 989–1008.
25. Voos, H. Nonlinear and neural network-based control of a small four-rotor aerial robot. In Proceedings of the 2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Zurich, Switzerland, 4–7 September 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 1–6.
26. Cloutier, J.R. State-dependent Riccati equation techniques: An overview. In Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041), Albuquerque, NM, USA, 6 June 1997; IEEE: Piscataway, NJ, USA, 1997; Volume 2, pp. 932–936.
27. Dierks, T.; Jagannathan, S. Output feedback control of a quadrotor UAV using neural networks. *IEEE Trans. Neural Netw.* **2009**, *21*, 50–66.
28. Jafari, M.; Xu, H. Intelligent control for unmanned aerial systems with system uncertainties and disturbances using artificial neural network. *Drones* **2018**, *2*, 30. [CrossRef]
29. Jiang, B.; Li, B.; Zhou, W.; Lo, L.Y.; Chen, C.K.; Wen, C.Y. Neural network based model predictive control for a quadrotor UAV. *Aerospace* **2022**, *9*, 460. [CrossRef]
30. Rieke, F.; Warland, D.; de Ruyter van Steveninck, R.; Bialek, W. *Spikes: Exploring the Neural Code*; MIT Press: Cambridge, MA, USA, 1999.
31. Thorpe, S.; Delorme, A.; Van Rullen, R. Spike-based strategies for rapid processing. *Neural Netw.* **2001**, *14*, 715–725. [PubMed]
32. Kampakis, S. Improved Izhikevich neurons for spiking neural networks. *Soft Comput.* **2012**, *16*, 943–953.
33. Yang, S.; Wang, J.; Deng, B.; Liu, C.; Li, H.; Fietkiewicz, C.; Loparo, K.A. Real-time neuromorphic system for large-scale conductance-based spiking neural networks. *IEEE Trans. Cybern.* **2018**, *49*, 2490–2503. [PubMed]
34. Farsa, E.Z.; Ahmadi, A.; Maleki, M.A.; Gholami, M.; Rad, H.N. A low-cost high-speed neuromorphic hardware based on spiking neural network. *IEEE Trans. Circuits Syst. II Express Briefs* **2019**, *66*, 1582–1586.
35. Ponulak, F.; Kasinski, A. Introduction to spiking neural networks: Information processing, learning and applications. *Acta Neurobiol. Exp.* **2011**, *71*, 409–433.
36. Abiyev, R.H.; Kaynak, O.; Oniz, Y. Spiking neural networks for identification and control of dynamic plants. In Proceedings of the 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Kaohsiung, Taiwan, 11–14 July 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 1030–1035.

37. Yamazaki, K.; Vo-Ho, V.K.; Bulsara, D.; Le, N. Spiking neural networks and their applications: A review. *Brain Sci.* **2022**, *12*, 863. [CrossRef] [PubMed]
38. Zhou, S.; Chen, Y.; Li, X.; Sanyal, A. Deep scnn-based real-time object detection for self-driving vehicles using lidar temporal data. *IEEE Access* **2020**, *8*, 76903–76912.
39. Kim, S.; Park, S.; Na, B.; Yoon, S. Spiking-yolo: Spiking neural network for energy-efficient object detection. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 11270–11277.
40. Qu, J.; Gao, Z.; Zhang, T.; Lu, Y.; Tang, H.; Qiao, H. Spiking Neural Network for Ultralow-Latency and High-Accurate Object Detection. *IEEE Trans. Neural Netw. Learn. Syst.* 2024, *early access.*
41. Bing, Z.; Meschede, C.; Röhrbein, F.; Huang, K.; Knoll, A.C. A survey of robotics control based on learning-inspired spiking neural networks. *Front. Neurorobot.* **2018**, *12*, 35.
42. Bohte, S.; Kok, J.; Poutre, J.L. Errorbackpropagation in Temporally Encoded Networks of Spiking Neurons. *Neurocomputing* **2002**, *48*, 17–37.
43. McKennoch, S.; Liu, D.; Bushnell, L. Fast Modifications of the SpikeProp Algorithm. In Proceedings of the International Joint Conference on Neural Networks, Vancouver, BC, Canada, 16–21 July 2006; pp. 3970–3977.
44. Zenke, F.; Ganguli, S. Superspike: Supervised learning in multilayer spiking neural networks. *Neural Comput.* **2018**, *30*, 1514–1541.
45. Lyapunov, A.M. The General Problem of the Stability of Motion. Ph.D. Thesis, Kharkov University, Kharkiv, Ukraine, 1892.
46. Gerstner, W.; Kistler, W.M. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*; Cambridge University Press: Cambridge, UK, 2002.
47. Luo, X.; Qu, H.; Wang, Y.; Yi, Z.; Zhang, J.; Zhang, M. Supervised learning in multilayer spiking neural networks with spike temporal error backpropagation. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 10141–10153.
48. Edwards, C.; Spurgeon, S. *Sliding Mode Control: Theory and Applications*; CRC Press: Boca Raton, FL, USA, 1998.
49. Respall, V.M.; Sellami, S.; Afanasyev, I. Implementation of autonomous visual detection, tracking and landing for AR. Drone 2.0 quadcopter. In Proceedings of the 2019 12th International Conference on Developments in eSystems Engineering (DeSE), Kazan, Russia, 7–10 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 477–482.
50. Santana, L.V.; Brandão, A.S.; Sarcinelli-Filho, M. An open-source testbed for outdoor navigation with the AR. Drone quadrotor. *IEEE Syst. J.* **2020**, *15*, 3597–3608.
51. Jeurgens, N. Identification and control implementation of an AR. Drone 2.0. Master's Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2017.
52. Sun, Y. Modeling, Identification and Control of a Quad-Rotor Drone Using Low-Resolution Sensing. Ph.D. Thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 2012.
53. Li, Q. Grey-Box System Identification of a Quadrotor Unmanned Aerial Vehicle. Master's Thesis, Delft University of Technology, Delft, The Netherlands, 2014.
54. The MathWorks Inc. *MATLAB Version: 9.13.0 (R2022b)*; The MathWorks: Natick, MA, USA, 2022.
55. Lopez-Sanchez, I.; Moreno-Valenzuela, J. PID control of quadrotor UAVs: A survey. *Annu. Rev. Control* **2023**, *56*, 100900.
56. Hoblit, F.M. *Gust Loads on Aircraft: Concepts and Applications*; Aiaa: Reston, VA, USA, 1988.
57. Shao, X.; Meng, Q.; Liu, J.; Wang, H. RISE and disturbance compensation based trajectory tracking control for a quadrotor UAV without velocity measurements. *Aerosp. Sci. Technol.* **2018**, *74*, 145–159.
58. Maass, W.; Natschläger, T.; Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **2002**, *14*, 2531–2560.

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.