

Article

Tuning the Proportional–Integral–Derivative Control Parameters of Unmanned Aerial Vehicles Using Artificial Neural Networks for Point-to-Point Trajectory Approach

Burak Ulu ¹, Sertaç Savaş ¹, Ömer Faruk Ergin ², Banu Ulu ³, Ahmet Kırnap ^{1,*}, Mehmet Safa Bingöl ^{1,4} and Şahin Yıldırım ¹

¹ Department of Mechatronics Engineering, Erciyes University, 38039 Kayseri, Turkey; burakulu@erciyes.edu.tr (B.U.); sertacsavas@erciyes.edu.tr (S.S.); msbingol@ohu.edu.tr (M.S.B.); sahin@erciyes.edu.tr (Ş.Y.)

² Department of Mechanical Engineering, Erciyes University, 38039 Kayseri, Turkey; omer@erciyes.edu.tr

³ Department of Software Engineering, Kayseri University, 38030 Kayseri, Turkey; banuulu@kayseri.edu.tr

⁴ Department of Mechatronics Engineering, Nigde Omer Halisdemir University, 51240 Nigde, Turkey

* Correspondence: ahmetkimap@erciyes.edu.tr

Abstract: Nowadays, trajectory control is a significant issue for unmanned micro aerial vehicles (MAVs) due to large disturbances such as wind and storms. Trajectory control is typically implemented using a proportional–integral–derivative (PID) controller. In order to achieve high accuracy in trajectory tracking, it is essential to set the PID gain parameters to optimum values. For this reason, separate gain values are set for roll, pitch and yaw movements before autonomous flight in quadrotor systems. Traditionally, this adjustment is performed manually or automatically in autotune mode. Given the constraints of narrow orchard corridors, the use of manual or autotune mode is neither practical nor effective, as the quadrotor system has to fly in narrow apple orchard corridors covered with hail nets. These reasons require the development of an innovative solution specific to quadrotor vehicles designed for constrained areas such as apple orchards. This paper recognizes the need for effective trajectory control in quadrotors and proposes a novel neural network-based approach to tuning the optimal PID control parameters. This new approach not only improves trajectory control efficiency but also addresses the unique challenges posed by environments with constrained locational characteristics. Flight simulations using the proposed neural network models have demonstrated successful trajectory tracking performance and highlighted the superiority of the feed-forward back propagation network (FFBPN), especially in latitude tracking within 7.52745×10^{-5} RMSE trajectory error. Simulation results support the high performance of the proposed approach for the development of automatic flight capabilities in challenging environments.

Keywords: neural networks; trajectory control; unmanned aerial vehicles; autonomous navigation; agricultural technologies



Citation: Ulu, B.; Savaş, S.; Ergin, Ö.F.; Ulu, B.; Kırnap, A.; Bingöl, M.S.; Yıldırım, Ş. Tuning the Proportional–Integral–Derivative Control Parameters of Unmanned Aerial Vehicles Using Artificial Neural Networks for Point-to-Point Trajectory Approach. *Sensors* **2024**, *24*, 2752. <https://doi.org/10.3390/s24092752>

Academic Editor: Arturo de la Escalera Hueso

Received: 12 February 2024

Revised: 15 March 2024

Accepted: 19 March 2024

Published: 26 April 2024



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

1. Introduction

In recent years, autonomous unmanned aerial vehicles (UAVs) have been given significant attention due to their enormous potential applications in civil and military fields. UAVs appear as micro-aerial robots that support a sustainable environment, offer a contactless delivery option, or can be used for hobby purposes [1–5]. In addition, aerial robots can be used autonomously in many applications with cameras and equipment mounted on them, such as increasing agriculture productivity, determining product maturity, detecting diseases and agricultural spraying [6]. Autonomous aerial robots should use a robust controller to perform assigned tasks with higher accuracy on specified trajectories. Many different methods are applied to control autonomous aerial robots. Masse et al. used the linear quadratic regulator (LQR) method and structured an H_∞ synthesis method for aerial

robot control. The results show that the H_{∞} method performs better than the LQR method in windy conditions. However, it presented an overview based on mathematical models and the section on actual working conditions is not mentioned [7]. Perozzi et al. utilized sliding mode control for trajectory tracking of a quadrotor. Results of numerical experiments confirmed the sliding mode control's success in stabilizing the quadrotor under varying wind. Their study aimed to develop a control structure based on predicted wind; however, the variable conditions in the real environment were not taken into account [8]. Celen and Oniz used fuzzy logic and neural network controllers for trajectory tracking for a quadcopter. The results show that artificial neural networks (ANNs) provide stability and robustness to the drone system [9]. In addition to the mentioned control methods, control models such as model predictive control, backstepping control and model reference adaptive control are also used [10,11]. Another popular method used for drone control is the proportional–integral–derivative (PID) controller [12]. Tuning the gain parameters of the PID controller is important, although the most commonly used method for tuning these parameters is trial and error. This method is time consuming as Lee and Peng pointed out in their study. Therefore, different methods are needed for tuning parameters [13].

The importance of PID control lies in its simplicity, efficiency, and above all, how easy it is to implement. There are many PID control studies in the literature developed for trajectory control in UAVs [14]. In these studies, many of the solutions given in terms of PID control are created temporarily, meaning that priority is given to solving the task rather than providing an analysis that will reveal the limitations and advantages of the control strategy. Independent of these studies, the number of ANN-based approaches that have increased their effectiveness has been increasing rapidly, especially in recent years.

Wang et al. used RBF (radial basis function) neural network to tune the autonomous flight PID controller parameters of UAVs in adapting to different environments [15]. Gao et al. used NN-PID to control the attitude and position of a quadrotor. Simulation results show that their NN-PID algorithm is more robust than the PID algorithm [16]. In many studies, quadrotors exhibit complex nonlinear dynamics resulting from complex interactions between multiple rotors and their effects on both translational and rotational motion. PID controllers designed by classical methods for linear systems have difficulty in accurately modeling and handling the nonlinear behavior of quadrotors. Therefore, superior methods such as artificial intelligence are needed for tuning PID parameters.

In this study, an ANN-based PID control parameter adjustment method is proposed for the autonomous flying robot system developed to determine the yield in an apple orchard where autotune mode is not possible [17]. This system is controlled with three different PID control structures for roll, pitch and yaw movements. Thus, the adaptation of the quadrotor to linear PID parameters is facilitated by using nonlinear behavioral parameters in ANN training. For this purpose, the flying robot system, which has a script coded in Python, performed flight simulations with 200 different randomly determined PID parameter combinations in the Mission Planner simulation environment to which it is connected via SITL, and a data set with position data as input and PID parameters as output is created. Simulations performed on the trajectory used as a reference while creating the data set show that changing the PID parameters of the yaw control does not cause a significant change in the trajectory error. Therefore, a neural network predictor model for yaw control is not developed in this study. The data set obtained for roll and pitch controllers is used for models developed with three different ANN algorithms. As a result, test flights are carried out in the Mission Planner simulation environment of the quadrotor system with the PID gain parameters estimated by three different ANN models, and the performance and error values of the estimated PID parameters were observed. Simulation results are presented comparatively in tables and graphics.

2. Methodology

Using the conventional approach of autotune mode to establish control parameters for the quadrotor robotic system is deemed unsuitable, particularly in an apple orchard's narrow and sparsely covered corridors, as illustrated in Figure 1. Autotune mode is frequently used in studies to determine PID control parameters. Mini aerial vehicles using this mode automatically determine the optimum K_p , K_i and K_d gain parameters in large and open areas, with movements similar to the dance of bees in the air. However, it is not possible to apply the method in the apple orchard where the study will be carried out to determine the yield since the apple orchards have narrow corridors and the upper parts are covered with thin cover. For this reason, in this study, the K_p , K_d and K_i gain parameters are determined with artificial intelligence methods in the determined trajectories and the system performance is evaluated. In the study, the determination of K_p , K_d and K_i gain parameters is carried out in a simulation environment, and the trajectory tracking performance of the quadrotor system is examined with the obtained gain parameters.



Figure 1. Apple orchard where flying robotic system is used, Yeşilhisar, Kayseri, Turkey.

This section describes the developed system with a broad explanation. Firstly, software specifications are outlined in detail. The modified standard PID controller is described with tuned gain parameters. The general methodology of this article is given in Figure 2. The study, seen in Figure 2, consists of four basic stages.

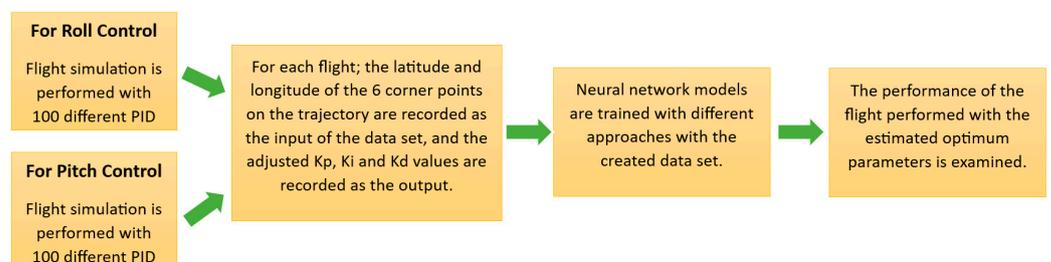


Figure 2. Flow chart description of the proposed methodology.

2.1. Software Specifications

Ardupilot software (Copter-4.4.0) is embedded in the controller to run flight commands and manage the flight. Through this software, precise control of the quadrotor can be achieved with PID control parameters determined through the flight planning program.

The high accuracy of the simulation software is demonstrated by comparing simulated and experimental trajectories for three different UAVs [18]. In this study, Mission Planner is used as the flight planning and simulation program. Furthermore, the quadrotor dynamics [19] required for flight are provided by embedded libraries in the Mission Planner open source software [20].

In addition to referenced studies, flight trajectory is created for simulation to compare trajectory tracking performances with different K_p , K_i and K_d parameters. For the quadrotor to track this trajectory, which is similar to the corridors in the apple orchard, a mission program is coded in Python using the Dronekit library. Software in the Loop (SITL) is used to simulate systems operating in real time [21]. Using this program, each simulation flight performed with different parameters is tracked on the map as shown in Figure 3, and location data is recorded. These data are then processed and a data set is created for training the neural network model.

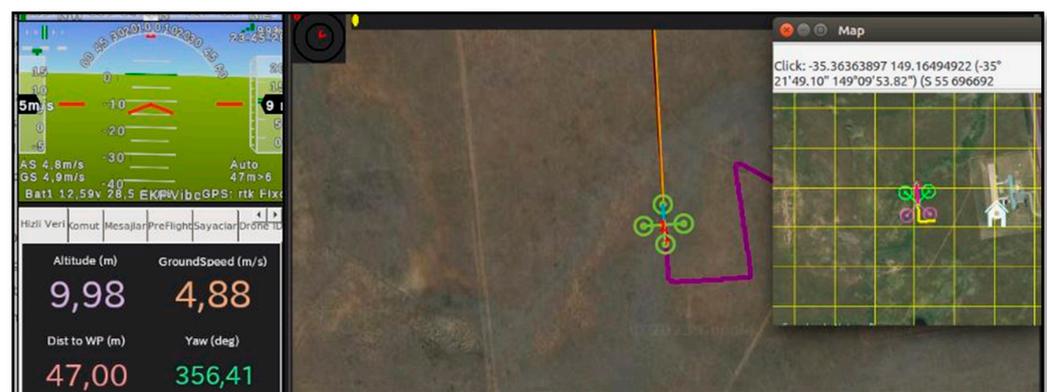


Figure 3. View of one of the simulation flights on the map.

2.2. Standard PID Controller System

PID controllers have been used for a wide variety of systems. Today, PID control is also used in many drone studies [22]. The continuous time PID controller is defined by Equation (1) [23]. To drive a plant output $y(t)$, towards a reference signal $r(t)$, the control input $u(t)$, is calculated by a closed loop PID controller, which uses the error in the output, that is, $e(t) = r(t) - y(t)$ [24]. In Equation (1), K_p is the proportional gain, K_i is the integral gain and K_d is the derivative gain.

$$u(t) = K_p \cdot e(t) + K_i \cdot \int e(t) dt + K_d \cdot \frac{d}{dt} e(t) \quad (1)$$

Appropriate coefficient values must be adjusted for the PID controller parameter to operate correctly. Although the Ziegler Nichols method [25] is the most widely used method for tuning these parameters in many systems, it is not easy to apply in such unique systems. Due to the high degree of non-linearity in these systems, it is difficult to modify the PID controller's gain parameters when there are external disturbances or the system parameters are changing. As a result, the PID controller's performance is reduced [26].

MAVs perform their flights according to specific modes. This study uses Auto and Guided modes for the quadrotor to perform the task fully autonomously. Auto mode is used when the flight program is uploaded to the controller, and Guided mode is used when it is managed externally through the program written with Dronekit. In both modes, position errors must be minimized by continuous monitoring of a controller so the quadrotor system can follow the determined trajectory. This system is controlled by three different PID control structures for roll, pitch and yaw movements. The PID control model of UAVs is shown schematically in Figure 4 for UAV movements.

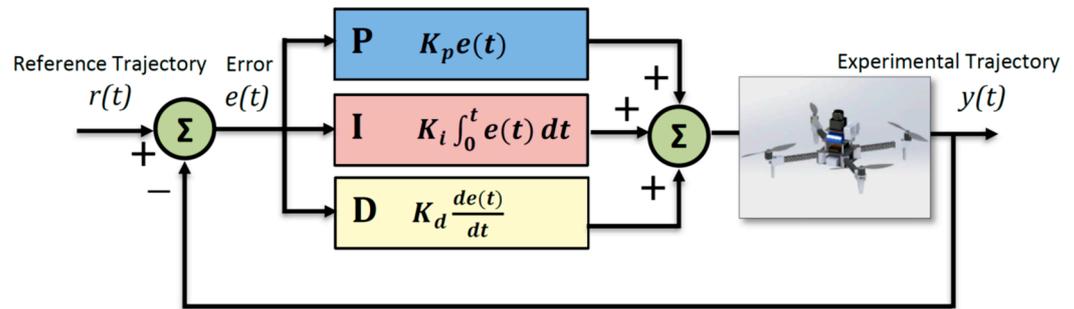


Figure 4. Standard PID control system.

K_p , K_i and K_d parameters must be tuned for each control model before the flight. This tuning can be performed by the trial-and-error method based on experience or adjusted autonomously in autotune mode, a mode developed for this type of MAV. If there is a suitable and sufficient area for flight, automatically tuning the PID controller gain parameters gives a more effective result. In this mode, the UAV moves autonomously in a wide-area trajectory, similar to the dance of bees, according to a determined algorithm. During this movement, K_p , K_i and K_d parameters are tuned in the most optimal way.

However, it is not possible to use the autotune mode because the micro-UAV designed in this study will perform its movement in a narrow and covered environment among trees. For this reason, an artificial neural network (ANN) model has been developed to determine the K_p , K_i and K_d parameters.

In the simulation environment, a trajectory was determined by reference to the row size of the tree array where the experimental study will be carried out. This trajectory is autonomously followed with datasets consisting of 100 different K_p , K_i and K_d parameters determined separately for the roll and pitch control models. During the sample preparation, it is experienced that 100 samples in the dataset are sufficient to extract the characteristics between the position-PID parameters. Changing the PID parameters of the yaw control does not cause a significant change in the trajectory error. Therefore, there is no need for an optimization for the yaw control model in the simulation flights, and the default values are used in all flights. The default values for the gain parameters are 0.2 for K_p , 0.02 for K_i and 0.002 for K_d .

In this method, 200 flight simulations are performed according to the parameter changes in two different control models. The K_p , K_i and K_d parameters adjusted in each simulation are used as output values for the samples in the training of the ANN model. These values are determined randomly through the program and the K_p , K_i and K_d parameters of one of the roll and pitch control models were changed while the others were kept constant. In this way, it is aimed to determine the optimum K_p , K_i and K_d control parameters for each movement type.

2.3. Proposed Artificial Neural Network (ANN)

For tuning PID parameters, several offline methods are available. In this section, the critical ideas of ANN, supervised learning and system identification methodologies are introduced and analyzed in relation to the theoretical development and applications of optimal PID controllers [27]. ANNs can offer flexible solutions to non-linear problems with their structure similar to the human brain.

ANNs where data flow only in the forward direction are feed-forward networks. ANNs with connections that allow data to flow both forward and backward are called feedback neural networks. One of the network structures used in this study is the feed-forward back propagation network (FFBPN). The FFBPN is a method of supervised learning. Radial basis neural network (RBNN) and cascade forward back propagation network (CFBPN), other network structures used in the study, are also supervised learning methods.

The schematic representation of the ANN model trained to estimate the K_p , K_i and K_d parameters for the FFBPN model is given in Figure 5.

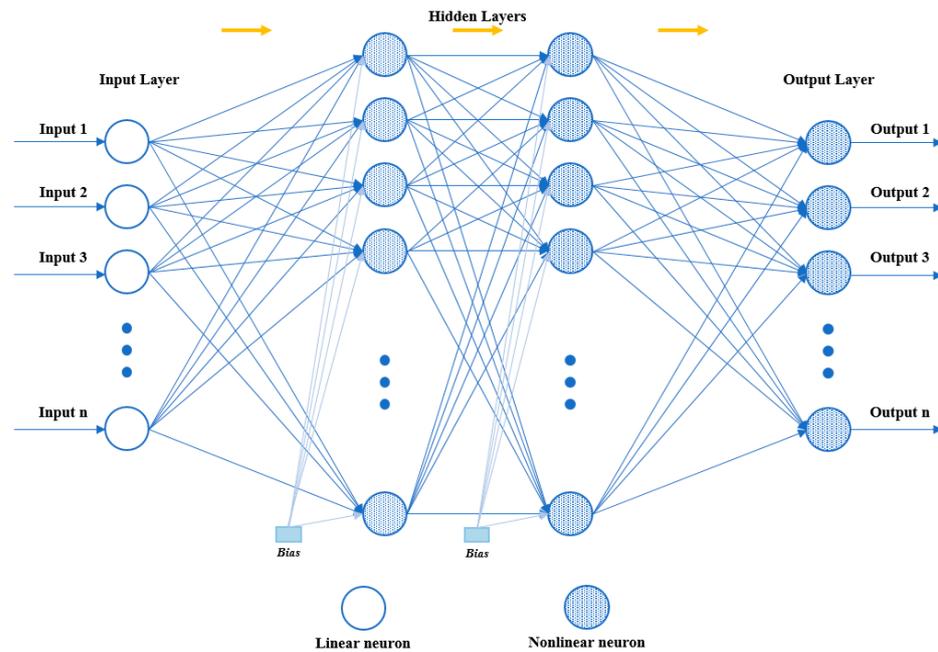


Figure 5. The schematic representation of the FFBNP model with signal flow direction.

The weights between input and hidden layers are updated as Equation (2) in FFBNP, where η is the learning rate and α is the momentum term. $E_2(t)$ is the propagation error between hidden and input layers. $E_1(t)$ is the error between experimental and neural network output signals [28,29].

$$\Delta W_{ij}(t) = -\eta \frac{\partial E_2(t)}{\partial W_{ij}} + \alpha \Delta W_{ij}(t-1) \quad (2)$$

The weights between the hidden and output layers are updated as Equation (3).

$$\Delta W_{jn}(t) = -\eta \frac{\partial E_1(t)}{\partial W_{jn}} + \alpha \Delta W_{jn}(t-1) \quad (3)$$

The CFBNP model is similar to a FFBNP model in using the backpropagation algorithm for weight updating. However, a fundamental characteristic of this network is that every layer of neurons is associated with all preceding layers of neurons. As other feed-forward networks, CFBNP has one or more interconnected hidden layers and activation functions. Each neuron has a bias of its own and each connection has a specific weight [30,31]. The schematic representation of the CFBNP model is given in Figure 6.

Each combination in the CFBNP learning sample (p_q, d_q) is calculated as follows [32]: p_q inputs are propagated forward through the layers of the m -layer neural network by Equation (4):

$$a^0 = p_q; a^k = f^k(W^k a^{k-1} - b^k), k = 1, \dots, M \quad (4)$$

where p_q is input, a is cell output, b is bias and w is weight.

Back propagate the sensitivities through the layers by Equation (5):

$$\delta^M = -2F'^M(n^M)(d_q - a^M); \delta^k = -F'^k(n^k)(W^{k+1T})\delta^{k+1}, k = M-1, \dots, 1 \quad (5)$$

Modify the biases and weights by Equations (6) and (7), respectively:

$$\Delta b^k = \eta \delta^k, k = 1, \dots, M \quad (6)$$

$$\Delta W^k = -\eta \delta^k (a^{k-1})^T, k = 1, \dots, M \quad (7)$$

These steps continue until the stopping criterion is reached.

The nonparametric estimation of multidimensional functions using sparse amounts of training data is a common application for RBNN. Fast and thorough training of RBNNs makes them effective [33,34]. RBNNs have many benefits, including strong global approximation ability, no local minimum difficulties and fast learning speed [35]. FFBPNs can have one or multiple hidden layers, while RBNNs have only one. The input layer, which is the first layer of the RBNN, just serves to transfer information and does not process the input data in any way. The second layer is a hidden layer. The third layer is the output layer, which will linearly transform the input data and then the output [36]. The schematic representation of the RBNN model is given in Figure 7.

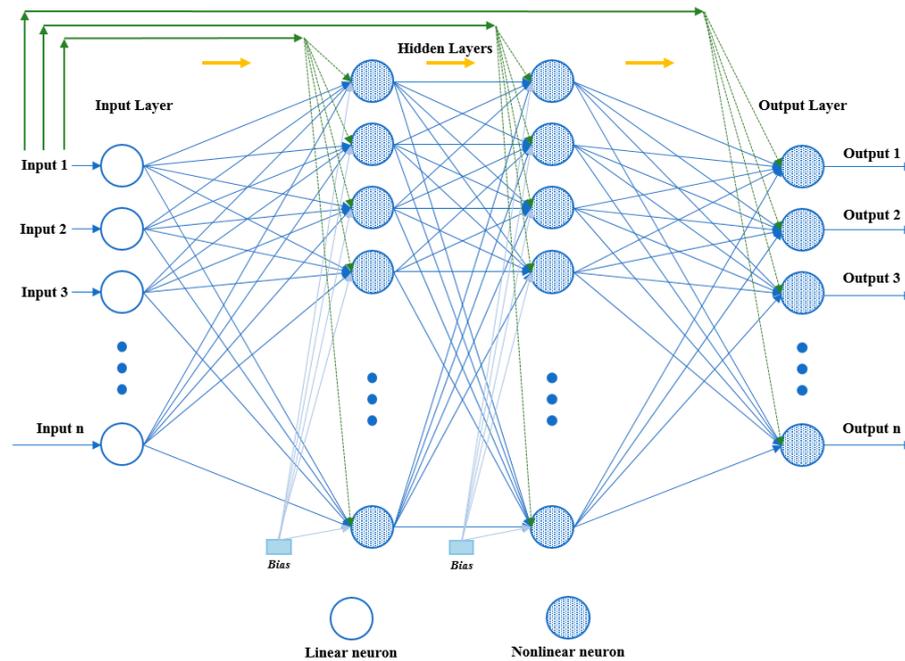


Figure 6. The schematic representation of the CFBPN model.

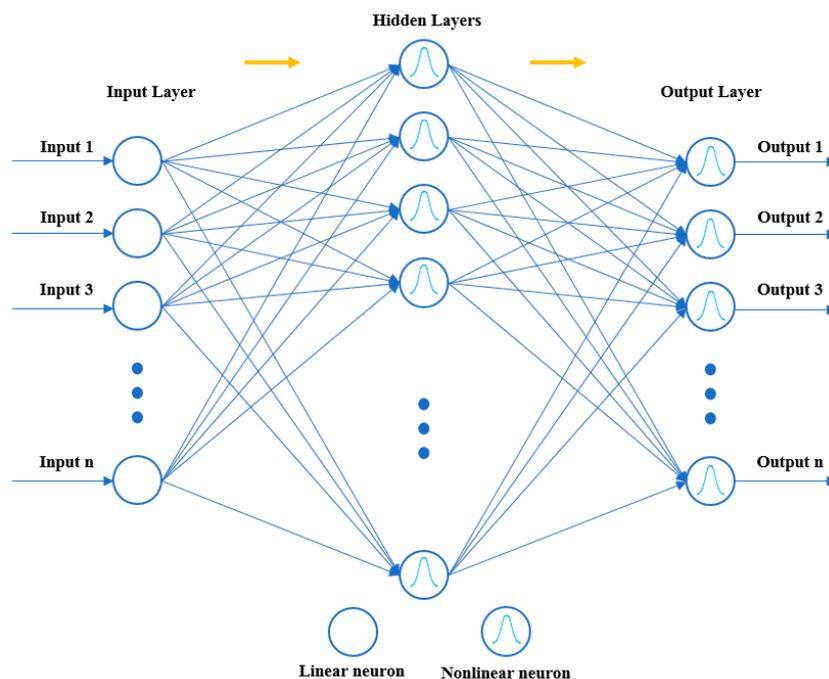


Figure 7. The schematic representation of the RBNN model.

The objective function in RBNN can be defined by Equation (8), where h_i is the height value of sample i [37].

$$E = \frac{1}{2} \sum_{i=1}^n (h_i - f(x_i))^2 \quad (8)$$

The $f(x)$ given in Equation (8) can be defined as in Equation (9).

$$f(x) = \sum_{i=1}^n \omega_i \phi(\|x - c_i\|) \quad (9)$$

where n is the number of neurons in the hidden layer, $\omega_i \in W$ is the weight of neuron i in the linear output neuron, c_i denotes the center vector of the neuron i , $\phi(\cdot)$ denotes the nonlinear function that is a multiquadratic function in Equation (10).

$$\phi(r) = r \sqrt{1 + \zeta^2} \quad (10)$$

where r denotes the distance between unknown and known data, and ζ is a smoothing factor between 0 and 1. As a result, the formula used to calculate the weights in the network structure is given in Equation (11).

$$W(t+1) = W(t) - \eta \cdot \frac{\partial E}{\partial W} = W(t) + \eta \cdot \sum_{i=1}^n (h_i - f(x_i)) \phi(\|x - c_i\|) \quad (11)$$

The latitude and longitude values (12 pieces) of the corner points determined from the position data recorded at 2 s intervals during the flight are used as input in the training of the ANN model. The output of the ANN model is the parameters K_p , K_i and K_d . The performance of the ANN models for the roll and pitch control models are examined in detail in the Results section.

3. Simulation Results

3.1. Estimation of PID Parameters

In the study, three different artificial neural network models, namely feed-forward back propagation neural network, cascade-forward back propagation neural network and radial basis neural network, are designed to determine the PID control parameters of the flying robot system. The designed models are trained with the trajectory points-PID gain values data set obtained in the Mission Planner simulation environment. Z-score normalization is applied to the entire data set for the training process. PID parameters for the reference trajectory are estimated with the trained artificial neural network models. The data set allocated for 100 rolling and 100 pitching values is divided into three groups as 70% train, 15% test and 15% validation for FFBN and CFBN. This separation is performed randomly among the data. Then, flight tests are carried out in the same simulation environment with the obtained parameters and flight performances are compared in terms of latitude, longitude and altitude.

FFBN and CFBN artificial neural network models are designed as twelve inputs, three outputs and two hidden layers with eight neurons each. In the training process of both models, three training algorithms, the Levenberg–Marquart (trainlm), scaled conjugate gradient (trainscg) and BFGS quasi-newton (trainbfg), and three activation functions (logsig, radbas and tansig) are used. The training algorithm and activation function that gave the lowest training error are determined by the grid search method to be used in performance tests. Training parameters of artificial neural network models are given in Table 1. MSE values of FFBN models and estimated PID parameters for the rolling and pitching controllers are given in Table 2. Figure 8 shows the performance plots of FFBN neural network models with the best training MSE values for rolling and pitching control, respectively.

MSE values of CFBN models and estimated PID parameters for the rolling and pitching controllers are given in Table 3. Figure 9 shows the performance plots of CFBN neural network models with the best training MSE values for rolling and pitching control.

Finally, the RBNN model is designed and training procedures are carried out to determine the control parameters of both control approaches. Although this artificial neural network model also has twelve inputs and three outputs, it has 100 neurons in its single hidden layer. The spread value of the model is selected as 0.1. Training MSE values and control gain values estimated with the trained network are given in Table 4. Additionally, the training performance graphs of RBNN network trained for rolling and pitching control are shown in Figure 10.

Table 1. FFBN and CFBN training parameters.

Performance Function	MSE
Maximum Number of Iterations (Epochs)	1000
Minimum Gradient	10^{-6}
Damping Factor (μ) (trainlm)	0.15

Table 2. MSE values of FFBN models and estimated PID parameters for the rolling and pitching controllers.

Control Type	Training Algorithm	Activation Function	MSE	K_p	K_i	K_d	
Rolling Control	Levenberg–Marquardt	tansig	0.9077	2.1101	2.0188	0.5137	
		radbas	0.8778	2.8903	1.9165	0.7645	
		logsig	0.9184	2.0596	2.0874	0.5805	
	Scaled Conjugate Gradient	tansig	0.9051	2.2497	1.3240	0.4064	
		radbas	1.3643	1.9666	2.3850	0.2320	
		logsig	0.9533	2.0811	1.8916	0.5350	
	BFGS Quasi-Newton	tansig	0.9735	2.1330	2.0543	0.4461	
		radbas	0.8526	2.0328	2.2562	0.7115	
		logsig	0.9740	2.4656	2.3052	0.5739	
	Pitching Control	Levenberg–Marquardt	tansig	0.6101	2.4077	1.8095	0.6013
			radbas	0.7179	3.5663	2.0397	1.3567
			logsig	0.6539	1.9875	2.6991	0.8251
Scaled Conjugate Gradient		tansig	0.9665	1.7854	2.2571	0.4842	
		radbas	0.5881	2.7480	2.2292	0.8836	
		logsig	0.7993	2.8513	2.4859	0.4041	
BFGS Quasi-Newton		tansig	0.7017	2.4867	2.4330	1.1562	
		radbas	0.6019	1.4790	2.5836	0.5073	
		logsig	0.9760	2.3690	2.2313	0.7147	

Table 3. MSE values of CFBN models and estimated PID parameters for the rolling and pitching controllers.

Control Type	Training Algorithm	Activation Function	MSE	K_p	K_i	K_d
Roll Control	Levenberg–Marquardt	tansig	0.6911	1.7155	0.6107	1.1281
		radbas	0.9593	2.8376	1.3909	0.8655
		logsig	0.8521	2.5103	1.0876	0.6433

Table 3. Cont.

Control Type	Training Algorithm	Activation Function	MSE	K_p	K_i	K_d
Roll Control	Scaled Conjugate Gradient	tansig	0.9067	2.6215	1.5810	0.6680
		radbas	0.8592	2.4244	2.3351	0.6117
		logsig	0.9466	1.9622	2.2506	0.4117
	BFGS Quasi-Newton	tansig	0.8748	2.3739	1.5228	1.2769
		radbas	0.9827	1.6846	1.3402	0.4403
		logsig	0.9478	2.4340	1.1644	0.5635
Pitch Control	Levenberg–Marquardt	tansig	0.6539	2.5264	1.9099	1.7538
		radbas	0.4553	2.0911	2.5904	1.1806
		logsig	0.6549	1.4632	1.3870	1.1121
	Scaled Conjugate Gradient	tansig	0.5553	1.5742	2.1896	0.7747
		radbas	0.7048	2.3274	3.4386	1.7152
		logsig	0.6983	2.4532	1.6562	2.3723
BFGS Quasi-Newton	tansig	0.3947	1.2427	0.6283	0.7927	
	radbas	0.6913	2.5531	1.9590	1.2485	
	logsig	0.7019	1.9238	1.2242	0.0080	

Table 4. MSE values of RBNN model and estimated PID parameters for the roll and pitch controllers.

Control Type	MSE	K_p	K_i	K_d
Rolling Control	0.0070	2.8264	3.2051	0.8674
Pitching Control	1.2665×10^{-30}	0.7937	1.7214	0.0803

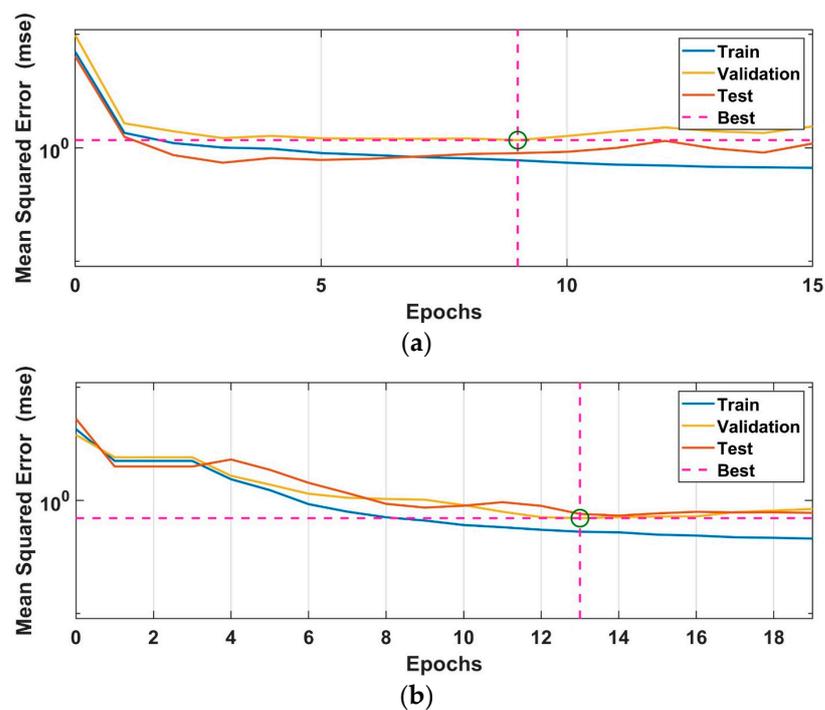


Figure 8. Training performance graphs of FFBPN models giving the best training results used to estimate PID parameters of the controllers for (a) rolling and (b) pitching.

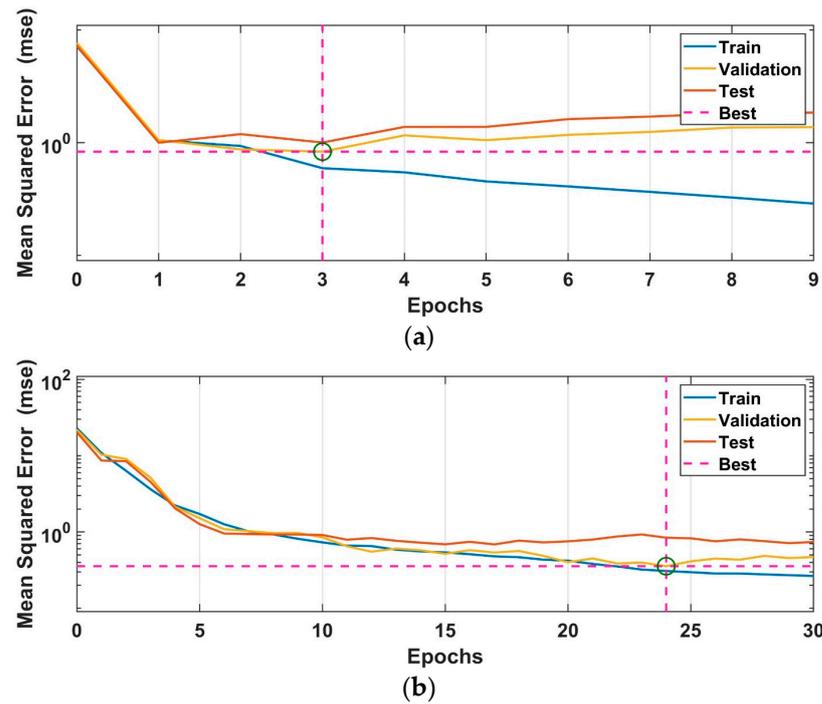


Figure 9. Training performance graphs of CFBPN models giving the best training results used to estimate PID parameters of the controllers for (a) roll control and (b) pitch control.

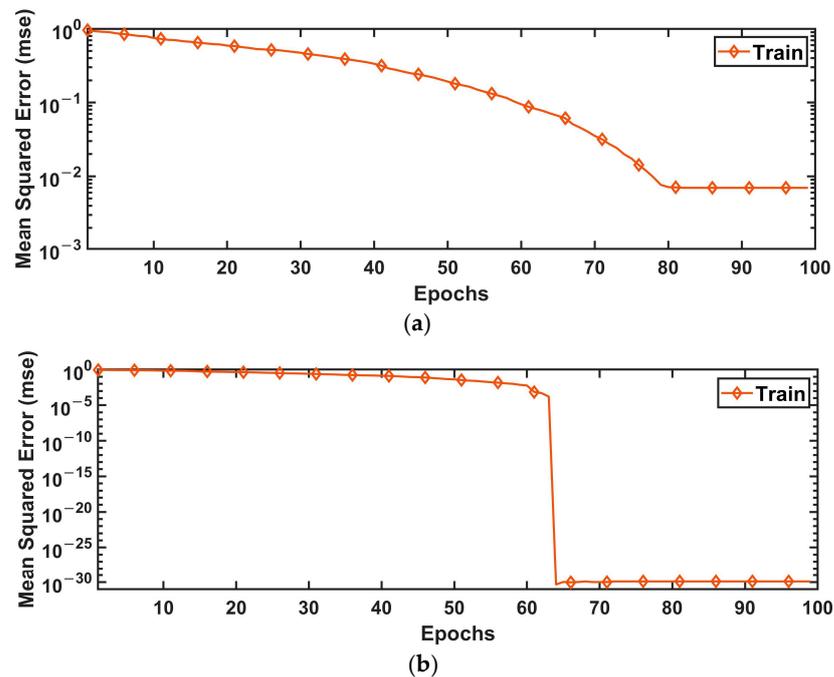


Figure 10. Training performance graphs of RBNN model used to estimate PID parameters of the controllers for (a) rolling and (b) pitching.

3.2. Flight Performance Tests

A reference flight trajectory is designed in the Mission Planner program to analyze the performance of PID control gain parameters obtained with artificial neural network models in roll and pitch control. Flights are carried out with each PID value and the tracking performance of the reference latitude, longitude and altitude values are analyzed with four error metrics. The mathematical expressions of the error metrics RMSE (root mean square

error), MSE (mean square error), MAE (mean absolute error) and MAPE (mean absolute percentage error) used for this purpose are given in Equations (12)–(15). RMSE (root mean square error) is the square root of the mean of the squares of the differences between the actual and reference values, giving greater weight to larger errors. MSE (mean squared error) is the average of the squares of the differences between the actual and reference values, providing an average value according to the error squares. MAE weights all errors equally by averaging the absolute values of the error amounts, thus reducing the weight of larger errors. MAPE evaluates the relative accuracy of actual values relative to the reference by providing a percentage measure of errors, making it easier to compare data at different scales.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - r_i)^2} \quad (12)$$

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - r_i)^2 \quad (13)$$

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - r_i| \quad (14)$$

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - r_i}{y_i} \right| \cdot 100 \quad (15)$$

In the equations, n is the number of samples, y_i is the i th actual output value, and r_i is the i th reference value. The reference trajectory points determined for the performance test are given in Table 5 and the reference trajectory is given in Figure 11. Additionally, rolling and pitching PID controller gain values are given in Table 6 for all artificial neural network models.

Table 5. Latitude, longitude and altitude values of the reference trajectory points to be used for performance testing.

	P ₁	P ₂	P ₃	P ₄	P ₅	P ₆
Latitude (°)	−35.363157	−35.362557	−35.362557	−35.363157	−35.363157	−35.362557
Longitude (°)	149.163687	149.163687	149.162939	149.162938	149.162191	149.162190
Altitude (m)	3	3	3	3	3	3

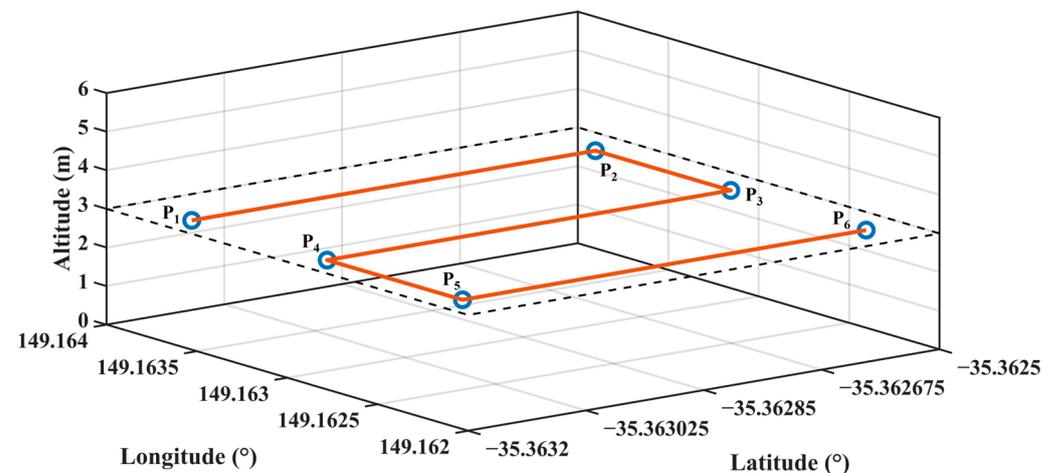
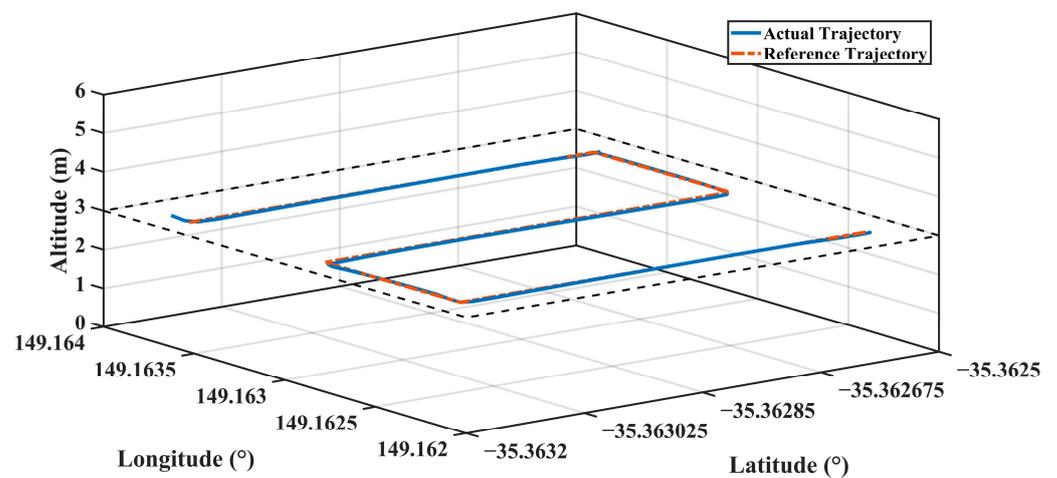
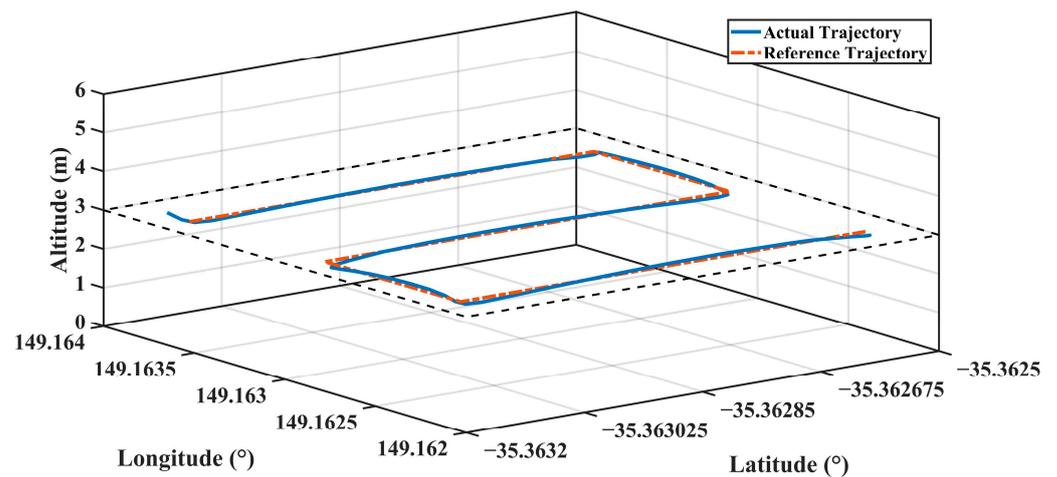


Figure 11. Reference trajectory and corner points (P₁: Start point, P₂–P₅: Trajectory points and P₆: End point) to be used for performance testing.

Table 6. Roll and pitch PID controller gain values for all artificial neural network models.

	Roll Control			Pitch Control		
	K_p	K_i	K_d	K_p	K_i	K_d
FFBPN	2.0328	2.2562	0.7115	2.7480	2.2292	0.8836
CFBPN	1.7155	0.6107	1.1281	1.2427	0.6283	0.7927
RBNN	2.8264	3.2051	0.8674	0.7937	1.7214	0.0803

The actual and reference trajectories of the flight realized with PID parameters estimated with FFBPN are given in Figure 12, CFBPN in Figure 13, RBNN in Figure 14. In addition, error metrics for latitude, longitude and altitude values, which are position information for each flight, are given in Table 7.

**Figure 12.** The actual and reference trajectory variations of the flight performed with PID parameters estimated with FFBPN.**Figure 13.** The actual and reference trajectory variations of the flight performed with PID parameters estimated with CFBPN.

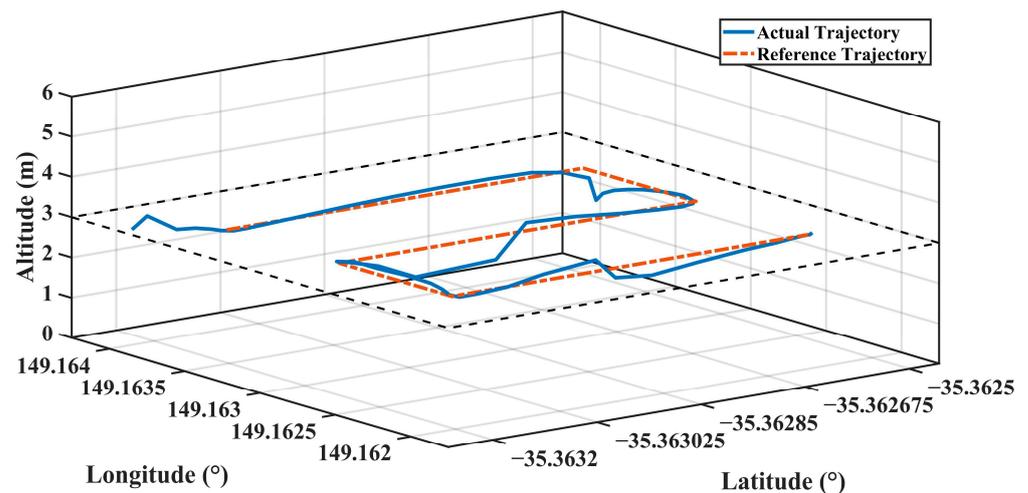


Figure 14. The actual and reference trajectory variations of the flight performed with PID parameters estimated with RBNN.

Table 7. Error metrics for latitude, longitude and altitude values.

NN Model	Coordinate	RMSE	MSE	MAE	MAPE
FFBPN	Latitude	7.52745×10^{-5}	5.66626×10^{-9}	4.90424×10^{-5}	8.16141×10^{-5}
	Longitude	0.00011	1.16874×10^{-8}	6.04816×10^{-5}	2.87345×10^{-5}
	Altitude	0.01878	0.00035	0.01422	0.25593
CFBPN	Latitude	7.57585×10^{-5}	5.73935×10^{-9}	4.90346×10^{-5}	6.97564×10^{-5}
	Longitude	0.00011	1.311565×10^{-8}	6.43695×10^{-5}	3.49091×10^{-5}
	Altitude	0.05800	0.003364	0.05081	0.10621
RBNN	Latitude	0.00013	1.658104×10^{-8}	8.98963×10^{-5}	2.22769×10^{-5}
	Longitude	0.00014	1.823412×10^{-8}	8.10415×10^{-5}	3.63931×10^{-5}
	Altitude	0.20918	0.043756	0.14407	1.62034

The reference and actual latitude, longitude and altitude values for the flights performed with all control parameters obtained with the three ANN models are shown in Figure 15, and the RMSE error graphs of these coordinate values are shown in Figure 16.

When the error metrics and graphs are examined, the results are quite close in flights where PID parameters obtained with FFBPN and CFBPN models are used. However, the performance of the parameters obtained with RBNN appears to be unsuccessful, especially in terms of latitude and altitude values. Especially, fluctuations in the flying robot system's altitude values cannot be ignored. Although the training error of the RBNN model is lower than the other two models, the fact that it is not successful in the test results reveals the possibility of overfitting. Therefore, it is understood that it cannot produce a robust response to external situations. This shows that the RBNN model is unsuitable for this specific task compared to the other two.

When FFBPN and CFBPN are compared with each other, it is seen that FFBPN parameters provide more successful tracking. Again, although the train errors of these models are quite close to each other, the test results reveal that the train performance should be evaluated in terms of its own hyperparameters within each network model. When the flight performance of the parameters estimated by the two models is examined, the models do not reveal very different results in terms of latitude and longitude values. In contrast, in the flight study conducted with CFBPN, it is observed that there is an increase in altitude on long straight roads and a decrease in corner points. This shows that the FFBPN model is more successful than others in predicting altitude changes.

Additionally, feed-forward networks include only forward connections between layers, whereas cascade-forward networks also have direct connections of the inputs and outputs of each layer to subsequent layers. While this enables cascade-forward networks to model more complex relationships by capturing longer-range dependencies in the data set, it may cause undesirable lengths of training time and the risk of overfitting in non-complex data sets. Therefore, it is necessary to consider each artificial neural network model by assessing the nature and properties of the data set.

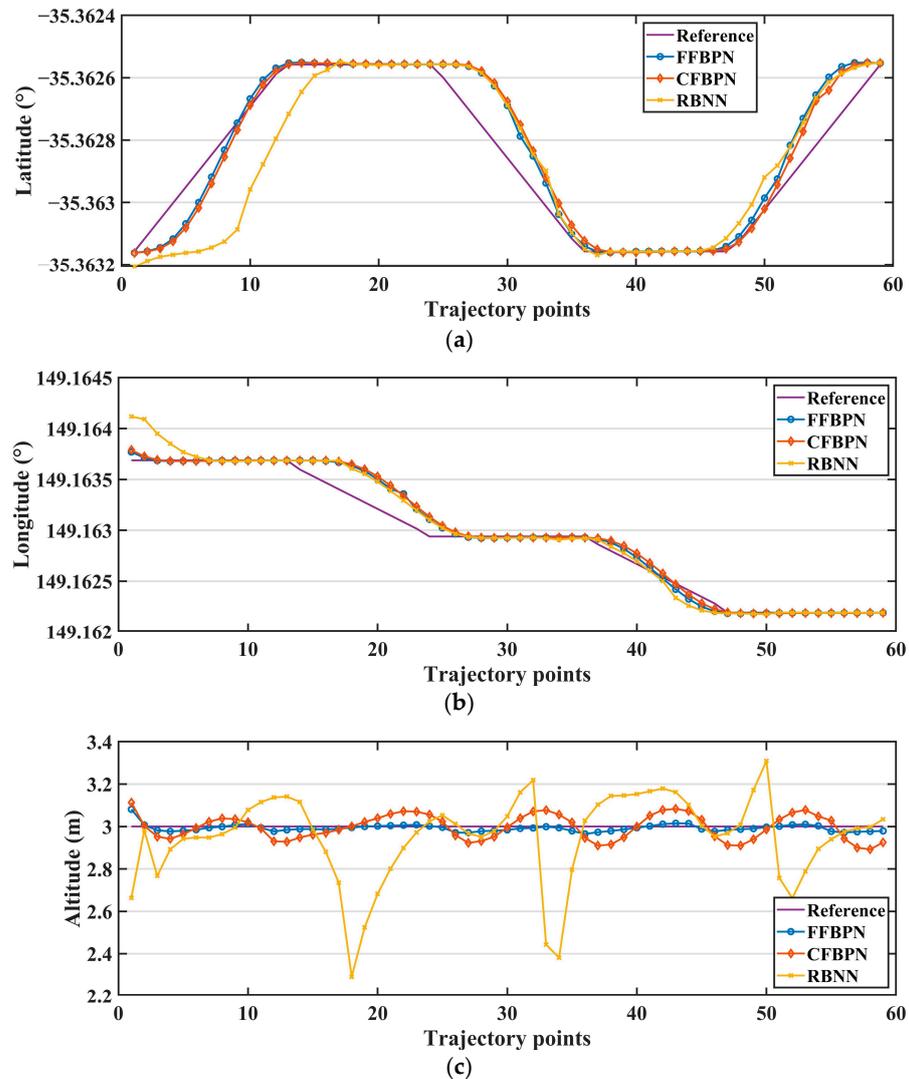


Figure 15. The reference and actual (a) latitude, (b) longitude and (c) altitude values.

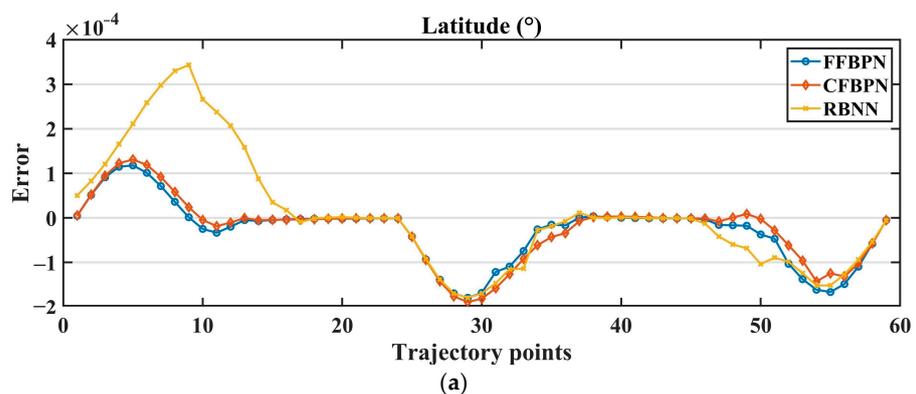


Figure 16. Cont.

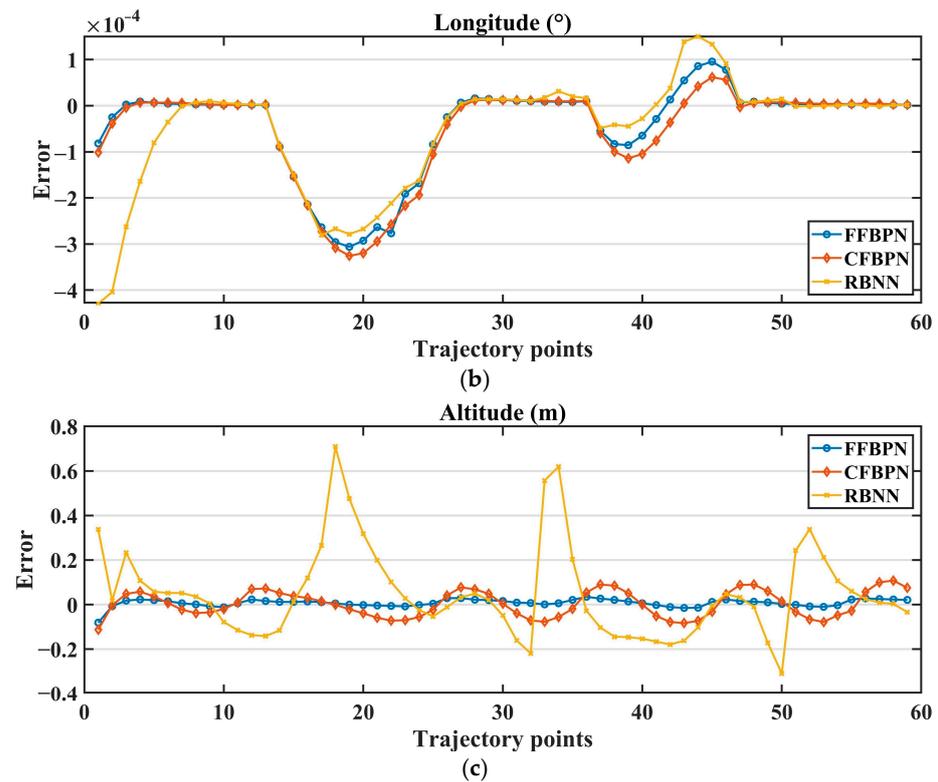


Figure 16. The RMSE error graphs of (a) latitude, (b) longitude and (c) altitude values.

4. Discussion and Conclusions

In this study, the optimum PID gain parameters of the micro air vehicle planned to fly in narrow apple orchard corridors are estimated with ANN. For this purpose, a data set is created from latitude, longitude and altitude data of flights performed with PID parameters randomly determined between significant lower and upper values. Since the realization of flights with random PID parameters in the real environment involves high risks, the flights are carried out in a simulation environment and artificial neural network models that make predictions based on the position data followed by the micro aerial vehicle according to parameter changes are developed with the obtained data set. Three different neural network models are designed with FFBPN, CFBPNN and RBNN architectures, which are frequently used to determine the optimum PID gain parameters. The neural network models are trained with different parameters and their performances are analyzed. The MSE values obtained show significant differences between the models. In addition, in order to test the success of the developed neural network models, simulation flights are performed using the PID gain parameters obtained with the models showing the best training performance. In the flight simulations, both FFBPN and CFBPNN models exhibited successful trajectory tracking performance in relation to their training performance, while FFBPN exhibited superior altitude tracking capabilities. As a result, it has been demonstrated that the PID parameters of a micro aerial vehicle can be tuned with artificial neural networks instead of randomizing them and an alternative solution has been proposed for challenging conditions where autotune mode cannot be used.

Author Contributions: B.U. (Burak Ulu) performed the simulation studies. S.S. and M.S.B. trained artificial neural network models and analyzed the results. Ö.F.E. and Ş.Y. took part in the preparation and evaluation of the data. B.U. (Banu Ulu) contributed to the development of the software used in the study. A.K. led and guided the research in the evaluation and interpretation of the results. All authors participated in the writing of the article. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

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

References

1. Almeida, D.R.A.; Broadbent, E.N.; Zambrano, A.M.A.; Wilkinson, B.E.; Ferreira, M.E.; Chazdon, R.; Meli, P.; Gorgens, E.B.; Silva, C.A.; Stark, S.C.; et al. Monitoring the structure of forest restoration plantations with a drone-lidar system. *Int. J. Appl. Earth Obs. Geoinf.* **2019**, *79*, 192–198. [\[CrossRef\]](#)
2. Mohsan, S.A.H.; Khan, M.A.; Noor, F.; Ullah, I.; Alsharif, M.H. Towards the Unmanned Aerial Vehicles (UAVs): A Comprehensive Review. *Drones* **2022**, *6*, 147. [\[CrossRef\]](#)
3. Kim, I.-H.; Jeon, H.; Baek, S.-C.; Hong, W.-H.; Jung, H.-J. Application of Crack Identification Techniques for an Aging Concrete Bridge Inspection Using an Unmanned Aerial Vehicle. *Sensors* **2018**, *18*, 1881. [\[CrossRef\]](#)
4. Chiang, W.-C.; Li, Y.; Shang, J.; Urban, T.L. Impact of drone delivery on sustainability and cost: Realizing the UAV potential through vehicle routing optimization. *Appl. Energy* **2019**, *242*, 1164–1175. [\[CrossRef\]](#)
5. Foehn, P.; Brescianini, D.; Kaufmann, E.; Cieslewski, T.; Gehrig, M.; Muglikar, M.; Scaramuzza, D. AlphaPilot: Autonomous drone racing. *Auton. Robot.* **2022**, *46*, 307–320. [\[CrossRef\]](#)
6. Tao, H.; Feng, H.; Xu, L.; Miao, M.; Long, H.; Yue, J.; Li, Z.; Yang, G.; Yang, X.; Fan, L. Estimation of Crop Growth Parameters Using UAV-Based Hyperspectral Remote Sensing Data. *Sensors* **2020**, *20*, 1296. [\[CrossRef\]](#)
7. MassÉ, C.; Gougeon, O.; Nguyen, D.-T.; SaussiÉ, D. Modeling and Control of a Quadcopter Flying in a Wind Field: A Comparison Between LQR and Structured H_∞ Control Techniques. In Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, USA, 12–15 June 2018; pp. 1408–1417.
8. Perozzi, G.; Efimov, D.; Biannic, J.-M.; Planckaert, L. Trajectory tracking for a quadrotor under wind perturbations: Sliding mode control with state-dependent gains. *J. Frankl. Inst.* **2018**, *355*, 4809–4838. [\[CrossRef\]](#)
9. Celen, B.; Oniz, Y. Trajectory Tracking of a Quadcopter Using Fuzzy Logic and Neural Network Controllers. In Proceedings of the 6th International Conference on Control Engineering & Information Technology (CEIT), Istanbul, Turkey, 25–27 October 2018; pp. 1–6.
10. Wei, P.; Chan, S.N.; Lee, S.; Kong, Z. Mitigating ground effect on mini quadcopters with model reference adaptive control. *Int. J. Intell. Robot. Appl.* **2019**, *3*, 283–297. [\[CrossRef\]](#)
11. Rothe, J.; Zevering, J.; Strohmeier, M.; Montenegro, S. A Modified Model Reference Adaptive Controller (M-MRAC) Using an Updated MIT-Rule for the Altitude of a UAV. *Electronics* **2020**, *9*, 1104. [\[CrossRef\]](#)
12. Pérez, I.C.; Flores-Araiza, D.; Fortoul-Díaz, J.A.; Maximo, R.; Gonzalez-Hernandez, H.G. Identification and PID control for a quadcopter. In Proceedings of the International Conference on Electronics, Communications and Computers (CONIELECOMP), Cholula, Mexico, 26–28 February 2014; pp. 77–82.
13. Lee, C.L.; Peng, C.C. Analytic Time Domain Specifications PID Controller Design for a Class of 2nd Order Linear Systems: A Genetic Algorithm Method. *IEEE Access* **2021**, *9*, 99266–99275. [\[CrossRef\]](#)
14. Oersted, H.; Ma, Y. Review of PID Controller Applications for UAVs. *arXiv* **2023**, arXiv:2311.06809. [\[CrossRef\]](#)
15. Wang, S.; Li, B.; Geng, Q. Research of RBF neural network PID control algorithm for longitudinal channel control of small UAV. In Proceedings of the 10th IEEE International Conference on Control and Automation (ICCA), Hangzhou, China, 12–14 June 2013; pp. 1824–1827.
16. Gao, W.N.; Fan, J.L.; Li, Y.N. Research on Neural Network PID Control Algorithm for a Quadrotor. *Appl. Mech. Mater.* **2015**, *719–720*, 346–351. [\[CrossRef\]](#)
17. Yıldırım, Ş.; Ulu, B. Deep Learning Based Apples Counting for Yield Forecast Using Proposed Flying Robotic System. *Sensors* **2023**, *23*, 6171. [\[CrossRef\]](#)
18. Yıldırım, Ş.; Çabuk, N.; Bakırcioğlu, V. Experimentally flight performances comparison of octocopter, decacopter and dodecacopter using universal UAV. *Measurement* **2023**, *213*, 112689. [\[CrossRef\]](#)
19. Asadi, D. Partial engine fault detection and control of a Quadrotor considering model uncertainty. *Turk. J. Eng.* **2022**, *6*, 106–117. [\[CrossRef\]](#)
20. Karachalios, T.; Moschos, P.; Orphanoudakis, T. Maritime Emission Monitoring: Development and Testing of a UAV-Based Real-Time Wind Sensing Mission Planner Module. *Sensors* **2024**, *24*, 950. [\[CrossRef\]](#)
21. Khaneghaei, M.; Asadi, D.; Tutsoy, Ö. Software in the Loop (SIL) Simulation for an Autonomous Multirotor Flight Planning and Landing with ROS and Gazebo. In Proceedings of the 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS), Istanbul, Turkey, 23–25 November 2023; pp. 1–10.
22. Noordin, A.; Mohd Basri, M.A.; Mohamed, Z. Real-Time Implementation of an Adaptive PID Controller for the Quadrotor MAV Embedded Flight Control System. *Aerospace* **2023**, *10*, 59. [\[CrossRef\]](#)

23. Sánchez-Palma, J.; Ordoñez-Ávila, J.L. A PID Control Algorithm with Adaptive Tuning Using Continuous Artificial Hydrocarbon Networks for a Two-Tank System. *IEEE Access* **2022**, *10*, 114694–114710. [[CrossRef](#)]
24. Pal, A.K.; Nestorović, T. Artificial Intelligence Neural Network Approach to Self Tuning of a Discrete-Time PID Control System. In Proceedings of the 9th International Conference on Systems and Control (ICSC), Caen, France, 24–26 November 2021; pp. 146–151.
25. Rodríguez-Abreo, O.; Rodríguez-Reséndiz, J.; Fuentes-Silva, C.; Hernández-Alvarado, R.; Falcón, M.D.C.P.T. Self-Tuning Neural Network PID with Dynamic Response Control. *IEEE Access* **2021**, *9*, 65206–65215. [[CrossRef](#)]
26. Bari, S.; Hamdani, S.S.Z.; Khan, H.U.; Rehman, M.u.; Khan, H. Artificial Neural Network Based Self-Tuned PID Controller for Flight Control of Quadcopter. In Proceedings of the International Conference on Engineering and Emerging Technologies (ICEET), Lahore, Pakistan, 21–22 February 2019; pp. 1–5.
27. Gómez-Avila, J.; López-Franco, C.; Alanis, A.Y.; Arana-Daniel, N. Control of Quadrotor using a Neural Network based PID. In Proceedings of the IEEE Latin American Conference on Computational Intelligence (LA-CCI), Guadalajara, Mexico, 7–9 November 2018; pp. 1–6.
28. Esim, E.; Yildirim, Ş. Drilling performance analysis of drill column machine using proposed neural networks. *Neural Comput. Appl.* **2017**, *28*, 79–90. [[CrossRef](#)]
29. Eski, I.; Erkaya, S.; Savas, S.; Yildirim, S. Fault detection on robot manipulators using artificial neural networks. *Robot. Comput.-Integr. Manuf.* **2011**, *27*, 115–123. [[CrossRef](#)]
30. Jesus, O.D.; Hagan, M.T. Backpropagation Algorithms for a Broad Class of Dynamic Networks. *IEEE Trans. Neural Netw.* **2007**, *18*, 14–27. [[CrossRef](#)] [[PubMed](#)]
31. Shohda, A.M.A.; Ali, M.A.M.; Ren, G.; Kim, J.-G.; Mohamed, M.A.-E.-H. Application of Cascade Forward Backpropagation Neural Networks for Selecting Mining Methods. *Sustainability* **2022**, *14*, 635. [[CrossRef](#)]
32. Tengelen, S.; Armand, N. Performance of Using Cascade Forward Back Propagation Neural Networks for Estimating Rain Parameters with Rain Drop Size Distribution. *Atmosphere* **2014**, *5*, 454–472. [[CrossRef](#)]
33. Loy, J. *Neural Network Projects with Python: The Ultimate Guide to Using Python to Explore the True Power of Neural Networks through Six Projects*; Packt Publishing Ltd.: Birmingham, UK, 2019.
34. Sohrabi, P.; Jodeiri Shokri, B.; Dehghani, H. Predicting coal price using time series methods and combination of radial basis function (RBF) neural network with time series. *Miner. Econ.* **2023**, *36*, 207–216. [[CrossRef](#)]
35. Deng, Y.; Zhou, X.; Shen, J.; Xiao, G.; Hong, H.; Lin, H.; Wu, F.; Liao, B.-Q. New methods based on back propagation (BP) and radial basis function (RBF) artificial neural networks (ANNs) for predicting the occurrence of haloketones in tap water. *Sci. Total Environ.* **2021**, *772*, 145534. [[CrossRef](#)]
36. Zhou, S.; Yang, C.; Su, Z.; Yu, P.; Jiao, J. An Aeromagnetic Compensation Algorithm Based on Radial Basis Function Artificial Neural Network. *Appl. Sci.* **2023**, *13*, 136. [[CrossRef](#)]
37. He, H.; Yan, Y.; Chen, T.; Cheng, P. Tree Height Estimation of Forest Plantation in Mountainous Terrain from Bare-Earth Points Using a DoG-Coupled Radial Basis Function Neural Network. *Remote Sens.* **2019**, *11*, 1271. [[CrossRef](#)]

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.