Novel Levenberg-Marquardt Based Learning Algorithm for Unmanned Aerial Vehicles

Andriy Sarabakha[a,b,*], Nursultan Imanberdiyev[a,b], Erdal Kayacan[a,*], Mojtaba Ahmadieh Khanesar[c], Hani Hagras[d]

[a]*School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore.*
[b]*ST Engineering-NTU Corp Laboratory, 50 Nanyang Avenue, 639798, Singapore.*
[c]*Faculty of Electrical and Computer Engineering, Semnan University, Semnan, 35131, Iran.*
[d]*The Computational Intelligence Centre, School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, United Kingdom.*

Abstract

In this paper, a Levenberg-Marquardt inspired sliding mode control theory based adaptation laws are proposed to train an intelligent fuzzy neural network controller for a quadrotor aircraft. The proposed controller is used to control and stabilize a quadrotor unmanned aerial vehicle in the presence of periodic wind gust. A proportional-derivative controller is firstly introduced based on which fuzzy neural network is able to learn the quadrotor's control model on-line. The proposed design allows handling uncertainties and lack of modelling at a computationally inexpensive cost. The parameter update rules of the learning algorithms are derived based on a Levenberg-Marquardt inspired approach, and the proof of the stability of two proposed control laws are verified by using the Lyapunov stability theory. In order to evaluate the performance of the proposed controllers extensive simulations and real-time experiments are conducted. The 3D trajectory tracking problem for a quadrotor is considered in the presence of time-varying wind conditions.

Key words: Fuzzy neural networks, sliding mode control, Levenberg-Marquardt algorithm, type-1 fuzzy logic control, unmanned aerial vehicle.

---

[*]Corresponding author

Email addresses: `andriy001@e.ntu.edu.sg` (Andriy Sarabakha), `iman0005@e.ntu.edu.sg` (Nursultan Imanberdiyev), `erdal@ntu.edu.sg` (Erdal Kayacan), `ahmadieh@semnan.ac.ir` (Mojtaba Ahmadieh Khanesar), `hani@essex.ac.uk` (Hani Hagras)

1. Introduction

The unmanned aerial vehicle's (UAV's) sector is becoming the fastest growing sector of the aerospace industry. Whether it is from increasing interest in civilian or military fields such as search and rescue operations (Sun et al., 2016), traffic surveillance (Zhou et al., 2015) and forest management (Wallace et al., 2014), UAVs have drawn tremendous attention from researchers and practitioners over the last few years. In particular, vertical take-off and landing (VTOL) UAVs have been the main interest of active research among the class of versatile flying robotic platforms due to their exceptional movement agility, relatively small size, and capability to hover and operate in cluttered environments. These advantages of VTOL UAV provide a cheap solution to dull, dirty and hazardous missions in many fields, such as target tracking (Vanegas et al., 2016), 3D terrain reconstruction (Torres et al., 2016), agriculture mapping (Perez-Ortiz et al., 2016), wildlife protection (Olivares-Mendez et al., 2015) and emergency evacuation (Sarabakha and Kayacan, 2016). In addition, apart from individual missions VTOL UAVs, e.g., quadrotors, have been widely exploited to perform cooperative tasks such as crop monitoring (Valente et al., 2011), assembly and structure construction (Alejo et al., 2014), and payload transportation (Michael et al., 2011).

However, the design of the flight control system is still a fundamental problem for multirotors, and for quadrotors in particular. Since quadrotor is an underactuated system and it has a constant fixed blade pitch on its propellers, its motion can only be achieved by altering the angular speed of each rotor. Thus, the controller design is not a straightforward task as its dynamics is highly nonlinear and underactuated. Furthermore, in typical surveillance operations, the designed controller should provide precise trajectory tracking capabilities to enhance UAV's flight safety and task survivability. However, the case of having uncertain working conditions and lack of modelling impose additional difficulties in the controller design. Hence, a robust and stable performance is required from designed controller regardless of unforeseen internal and external uncertainties.

There are two approaches to control the UAVs: model-based, which needs an exact model of the system, and model-free, which does not need an exact model of the system. Proportional-integral-derivative (PID) (Eresen et al., 2012), dynamic feedback linearization (DFL) (Hua et al., 2013), linear-quadratic regulator (LQR) (Dong et al., 2015), and model predictive control (MPC) (Hofer et al., 2016) are the examples of the most widely used model-based controllers. The aforementioned controllers deliver a good balance between implementation cost, control performance and operational complexities. On the other hand, the performance of PID and LQR is constrained by operating regions where nonlinearities are negligible, while in the case of MPC and DFL, a precise system model is required. However, in many UAV's applications, *a* *priori* knowledge about the process is just approximated. Therefore, the performance of these systems deteriorates because of the lack of modelling and both external and internal uncertainties.

The modelling of a highly complex nonlinear system is a time-consuming and challenging task. Therefore, as an alternative to the model-based control, model-free control may provide an intelligent control scheme. The fuzzy logic has an exceptional ability to handle the uncertainties in the system (Celikyilmaz and Turksen, 2009). Therefore, fuzzy logic controllers (FLCs) have become one of the most popular approaches to control nonlinear systems when their precise mathematical model is challenging to obtain (Castillo et al., 2016a; Cervantes and Castillo, 2015; Mendel et al., 2014). FLCs have been successfully designed and implemented to control mobile robots (Castillo et al., 2016b; Tai et al., 2016; Sanchez et al., 2015; Kumbasar and Hagras, 2014; Hagras, 2004), especially UAVs (Fu et al., 2016; Fakurian et al., 2014). However, one weakness of FLCs is that they need to be tuned to deal with uncertainties. Manual tuning can be a difficult and troublesome task. On the other hand, artificial neural networks (ANNs) are a family of supervised learning model that mimics the human brain. They are widely used in many applications due to their ability to learn from input-output data. The combination of FLC and ANN, called fuzzy neural network (FNN), fuses the reasoning ability of FLC to handle uncertain information with the training capability of ANN to learn from the controlled process (Gaxiola et al., 2015). FNN has shown promising results as it adopts the advantages from both FLC and ANN (Wang et al., 2015; Kim and Chwa, 2015; Gaxiola et al., 2014).

Derivative-based (computational approaches) and derivative-free (heuristic methods) are the two main classes of training algorithms for tuning the FNN parameters. The former require some partial derivatives to be calculated to tune the parameters of FNN, while the latter do not need derivative information in order to update the parameters of FNN. Gradient descent (GD) (Mukherjee and Routroy, 2012) and genetic algorithms (GAs) (Martinez-Martinez et al., 2015) are the most widely used approaches among the existing derivative-based and derivative-free training methods, respectively. However, GD training algorithms are based on the first-order Taylor expansion of a nonlinear function. Therefore, GD has low learning speed and mediocre efficacy, especially when the search space is complex or the solution is near to the local minimum. Besides, since GAs are based on a random search, they are slow-converging, non-deterministic and do not guarantee to find global maxima. Because of all aforementioned reasons, GD and GAs are not suitable for UAV applications. In order to obtain a more exact update law to optimize the parameters of a nonlinear function, second order Taylor expansion of the nonlinear function may be used which yields Newton's optimization algorithm. The implementation of Newton's optimization method is too difficult as it includes the calculation of Hessian matrix. To overcome these problems, more advanced algorithms have to be used such as sliding mode control (SMC) and Levenberg-Marquardt (LM) based algorithms. LM algorithm approximates the Hessian matrix which normally exists in newton's optimization method. This decreases the complexity of the algorithm considerably.

SMC is the robust control algorithm that can deal with the input uncertainties and external noise in a nonlinear system when it is controlled on the sliding surface (Hernandez-Gonzalez et al., 2012). FNN trained with SMC algorithm has been extensively used to control nonlinear systems due to its excellent properties, such as external disturbance rejection, parameter variation insensitivity and fast dynamic response (Ma et al., 2017; Lin et al., 2014). Moreover, over the last few years, the combination of SMC and FNN has been successfully applied in mechatronics engineering such as antilock braking system (Topalov et al., 2009) and robotic manipulator (Wai and Muthusamy, 2013).

On the other side, LM algorithm is a simplified and robust approximation of Newton's optimization algorithm which results in faster convergence than basic GD method while it is not as complex as Newton's optimization algorithm. In this case, the Hessian matrix is approximated and its complexity is greatly reduced. Thus, although it is more time consuming than GD, because of its superior performance, LM is preferred over GD for the consequent part parameters. LM algorithm has many successful implementations to tune the parameters of the FNN (Castillo et al., 2013; Salimifard and Safavi, 2013) FNN trained with LM algorithm exhibits not only faster-training speed and more robust capability, but also better forecasting accuracy (Khanesar and Kayacan, 2015). Furthermore, LM algorithm is one of the most successful training algorithms for small and medium sized patterns (Salim et al., 2013). Therefore, LM algorithm is a promising method for training FNNs.

UAVs are different from the most of commercial robotic systems, such as robot manipulators or ground vehicles, in a way that there are many custom-made UAV designs. Moreover, even if the UAV is commercialised one, many modifications can be done, such as additional on-board sensors or installation of different propellers, and the initial mathematical model of the system can not be used any-more through the control design. Therefore, our aim is to propose an adaptive and platform-free control scheme.

In this paper, feedback-error learning method is proposed in which type-1 FNN (T1FNN) is trained by a novel LM learning algorithm working in parallel with a conventional proportional-derivative (PD) controller for the quadrotor trajectory tracking problem under changing wind gust conditions. Thanks to learning capability of the proposed controllers, experimental results show the efficacy of the learning algorithms. The proposed T1FNN controller is superior to the conventional PD controller in terms of control accuracy. To the best of our knowledge, this is the first time these parameter update rules for T1FNN are tested in the real-time for quadrotor UAV.

The main contributions of this study are following:

- a novel LM theory-based learning algorithm with an adaptive learning rate for T1FNN is presented;
- the proof of the stability of the LM method for the training of T1FNN is proposed;
- for the first time, SMC and LM theory-based parameter update rules for T1FNN are implemented in robot operating system (ROS) using C++ to navigate a quadrotor UAV in real-time;
- the performances of proposed methods have been compared for the UAV 3D trajectory tracking problem in the presence of wind;
- an adaptive platform-free control of UAVs is proposed for the 3D tracking problem.

This paper is organised as follows. In Section 2, a nonlinear dynamical model of a quadrotor UAV is presented. In Section 3, the overall control scheme for a quadrotor is described. In Section 4, a fuzzy-neuro control approach with the parameter update rules is proposed. In Section 5, the efficiency of the control algorithm is tested in simulation. In Section 6, some real-time experimental tests are conducted in order to validate the proposed control solution. Finally, in Section 7, some conclusions and future work are drawn.

2. Mathematical Model of the Quadrotor UAV

Let $\vec{\mathbf{F}}_{\mathcal{W}} = \{\vec{\mathbf{x}}_{\mathcal{W}}, \vec{\mathbf{y}}_{\mathcal{W}}, \vec{\mathbf{z}}_{\mathcal{W}}\}$ be the world fixed frame (considered inertial under the hypothesis of flat and nonrotating Earth) and $\vec{\mathbf{F}}_{\mathcal{B}} = \{\vec{\mathbf{x}}_{\mathcal{B}}, \vec{\mathbf{y}}_{\mathcal{B}}, \vec{\mathbf{z}}_{\mathcal{B}}\}$ be the body frame. The origin of the body frame is located at the center of mass (COM) of the quadrotor, as illustrated in Fig. 1.

2.1. Rotor Dynamics

The four propellers rotations generate four forces (f_1, f_2, f_3, f_4), directed along the axis of rotation $\vec{\mathbf{z}}_{\mathcal{B}}$, and four torques ($\tau_1$, τ_2, τ_3, τ_4), around the axis of rotation $\vec{\mathbf{z}}_{\mathcal{B}}$. Let T be the total thrust which acts along $\vec{\mathbf{z}}_{\mathcal{B}}$ axis, whereas τ_ϕ, τ_θ and τ_ψ be the moments which act around $\vec{\mathbf{x}}_{\mathcal{B}}$, $\vec{\mathbf{y}}_{\mathcal{B}}$ and $\vec{\mathbf{z}}_{\mathcal{B}}$ axes, respectively. Under these considerations, the vector $\mathbf{u}$ of control inputs is chosen as in (Mahony et al., 2012):

$$\mathbf{u} = \begin{bmatrix} T & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T, \tag{1}$$

where l is the arm length.

2.2. Rotational Dynamics

The attitude of the quadrotor $\mathbf{o} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T$ is described by the three Euler's angles. The derivative with respect to time of the angles $\mathbf{o}$ is given by $\boldsymbol{\omega} = \begin{bmatrix}\dot{\phi} & \dot{\theta} & \dot{\psi}\end{bmatrix}^T$ and the angular velocity expressed in $\vec{\mathbf{F}}_{\mathcal{B}}$ is $\boldsymbol{\omega}_{\mathcal{B}} = \begin{bmatrix}p & q & r\end{bmatrix}^T$. The relation between $\boldsymbol{\omega}$ and $\boldsymbol{\omega}_{\mathcal{B}}$ is given by

$$\boldsymbol{\omega} = \mathbf{T}\boldsymbol{\omega}_{\mathcal{B}}, \tag{2}$$

where $\mathbf{T}$ is the transformation matrix from $\vec{\mathbf{F}}_{\mathcal{B}}$ to $\vec{\mathbf{F}}_{\mathcal{W}}$:

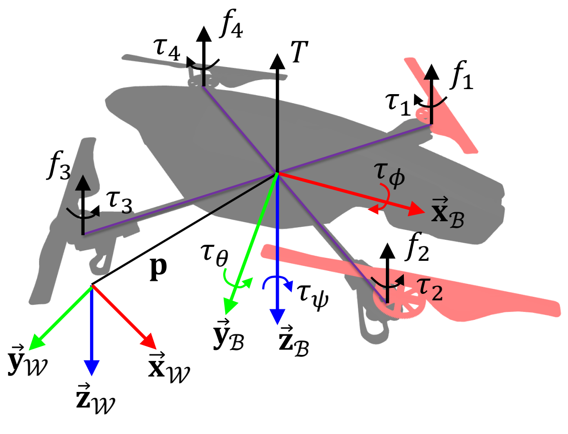
$$\mathbf{T} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}. \tag{3}$$

Using the Euler equation, the rotational dynamic equation is the following:

$$\mathbf{I}\dot{\boldsymbol{\omega}}_{\mathcal{B}} = -\boldsymbol{\omega}_{\mathcal{B}} \times \mathbf{I}\boldsymbol{\omega}_{\mathcal{B}} + \boldsymbol{\tau}, \tag{4}$$

where $\mathbf{I} = \mathrm{diag}(I_x, I_y, I_z)$ is the diagonal inertia matrix and $\boldsymbol{\tau} = \begin{bmatrix}\tau_\phi & \tau_\theta & \tau_\psi\end{bmatrix}^T$ is the vector of external torques. Using (2) and (4), the following rotational dynamic equations are obtained:

$$\begin{cases} \dot{\phi} = p + \sin\phi\tan\theta q + \cos\phi\tan\theta r & \dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{1}{I_x}\tau_\phi \\ \dot{\theta} = \cos\phi q - \sin\phi r & \dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{1}{I_y}\tau_\theta \\ \dot{\psi} = \frac{\sin\phi}{\cos\theta} q + \frac{\cos\phi}{\cos\theta} r & \dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{1}{I_z}\tau_\psi. \end{cases} \tag{5}$$



f_4
τ_4
T
f_1
τ_1
f_3
τ_3
τ_ϕ
$\vec{\mathbf{x}}_{\mathcal{B}}$
f_2
τ_2
τ_θ
$\mathbf{p}$
τ_ψ
$\vec{\mathbf{y}}_{\mathcal{B}}$
$\vec{\mathbf{z}}_{\mathcal{B}}$
$\vec{\mathbf{y}}_{\mathcal{W}}$
$\vec{\mathbf{x}}_{\mathcal{W}}$
$\vec{\mathbf{z}}_{\mathcal{W}}$

Figure 1: Quadrotor model with reference frames.

2.3. Translational Dynamics

The absolute position of the quadrotor $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is described by the three Cartesian coordinates of its COM in $\vec{\mathbf{F}}_{\mathcal{W}}$. The derivative with respect to time of the position $(x,\, y,\, z)$ is given by

$$\mathbf{v} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T = \begin{bmatrix} u & v & w \end{bmatrix}^T, \tag{6}$$

where $\mathbf{v}$ is the absolute linear velocity of the quadrotor's COM expressed with respect to $\vec{\mathbf{F}}_{\mathcal{W}}$. Let $\mathbf{v}_{\mathcal{B}} \in \mathbb{R}^3$ be the absolute linear velocity of the quadrotor expressed in $\vec{\mathbf{F}}_{\mathcal{B}}$. So, $\mathbf{v}$ and $\mathbf{v}_{\mathcal{B}}$ are related by

$$\mathbf{v} = \mathbf{R}\mathbf{v}_{\mathcal{B}}, \tag{7}$$

where $\mathbf{R} \in \mathsf{SO}(3)$ is the rotation matrix from $\vec{\mathbf{F}}_{\mathcal{B}}$ to $\vec{\mathbf{F}}_{\mathcal{W}}$:

$$\mathbf{R} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\phi\sin\theta - \cos\phi\sin\psi & \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\psi\sin\theta & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\phi\cos\theta \end{bmatrix}. \tag{8}$$

Using the Newton equation, the translational dynamic equation is

$$m\dot{\mathbf{v}} = \mathbf{F} \tag{9}$$

where m is the UAV's mass and $\mathbf{F}$ is the vector of external forces given by

$$\mathbf{F} = \begin{bmatrix} -\left(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi\right)T \\ -\left(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi\right)T \\ -\cos\phi\cos\theta T + mg \end{bmatrix}, \tag{10}$$

in which g is the gravitational acceleration. Using (6) and (9), the translational dynamic equations are as follows:

$$\begin{cases} \dot{x} = u & \dot{u} = -\frac{1}{m}\left(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi\right)T \\ \dot{y} = v & \dot{v} = -\frac{1}{m}\left(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi\right)T \\ \dot{z} = w & \dot{w} = -\frac{1}{m}\cos\phi\cos\theta T + g. \end{cases} \tag{11}$$

Finally, (5) and (11) provides the equations of the UAV's motion.

3. Control Scheme

The overall architecture of the controller is illustrated in Fig. 2. It consist of two interconnected control loops. The outer loop (position controller) is responsible for the quadrotor position tracking. While the inner loop (velocity controller) is responsible for the velocity tracking and attitude stabilisation. The quadrotor dynamical model is described in (5) and (11). Due to the quadrotor's underactuation, only four degrees of freedom (DOFs) out of its six DOFs can be controlled. In this paper, three positional coordinates of the quadrotor are chosen as the controlled variables. Therefore, the trajectory generator provides the desired position $\mathbf{p}^* = \begin{bmatrix} x^* & y^* & z^* \end{bmatrix}^T$.

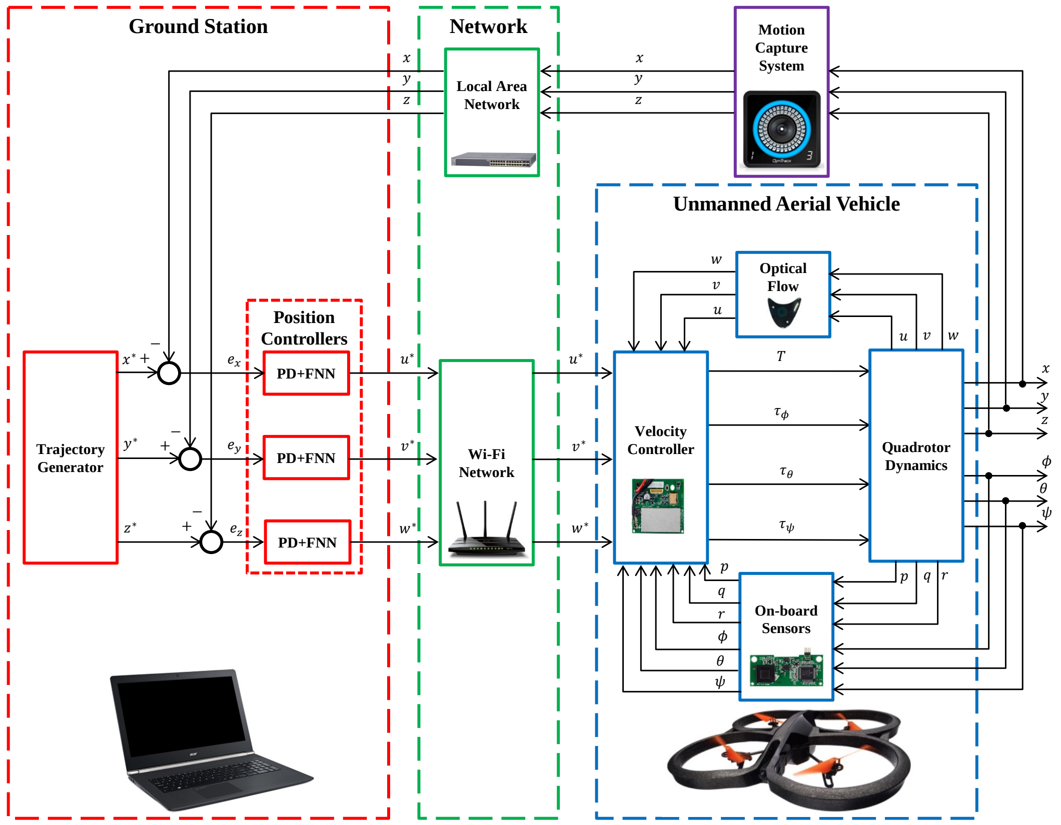


Figure 2: Block diagram of the control scheme for the quadrotor UAV.

3.1. *Position Controller*

The position controller consists of three independent controllers to track the desired position of the UAV on x, y and z axes. It takes three position error measurements from a position sensor, e.g., Mo-cap, GPS, SONAR, IR or vision-based:

$$\begin{cases} e_x = x^* - x \\ e_y = y^* - y \\ e_z = z^* - z, \end{cases} \tag{12}$$

and computes the desired velocity $\mathbf{v}^* = \begin{bmatrix} u^* & v^* & w^* \end{bmatrix}^T$ in order to reach the desired position $\mathbf{p}^*$. Three different position controllers (PD and two PD+FNN) are introduced in the next section.

3.2. *Velocity Controller*

For the velocity tracking, we use the nonlinear geometric controller on the special Euclidean group $\mathsf{SE}(3)$ (Lee et al., 2012). It receives the desired velocity $\mathbf{v}^*$ and using the actual velocity $\mathbf{v}$, actual attitude $\mathbf{o}$ and angular rate $\boldsymbol{\omega}_{\mathcal{B}}$, it computes the control input $\mathbf{u}$.

The desired direction of the first body axis is $\vec{\mathbf{x}}_{\mathcal{B}}^{*}$ and can be commanded by the higher-level controller. The desired direction of the third body axis is

$$\vec{\mathbf{z}}_{\mathcal{B}}^{*} = \frac{-k_{\mathbf{v}}\mathbf{e}_{\mathbf{v}} - mg\mathbf{e}_3}{\| - k_{\mathbf{v}}\mathbf{e}_{\mathbf{v}} - mg\mathbf{e}_3\|}, \tag{13}$$

where $\mathbf{e}_{\mathbf{v}} = \mathbf{v} - \mathbf{v}^{*}$ is the velocity error, $\mathbf{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ and $k_{\mathbf{v}}$ is some positive constant gain. Then, the desired direction of the second body axis can be computed as

$$\vec{\mathbf{y}}_{\mathcal{B}}^{*} = \frac{\vec{\mathbf{z}}_{\mathcal{B}}^{*} \times \vec{\mathbf{x}}_{\mathcal{B}}^{*}}{\|\vec{\mathbf{z}}_{\mathcal{B}}^{*} \times \vec{\mathbf{x}}_{\mathcal{B}}^{*}\|}. \tag{14}$$

In addition, $\vec{\mathbf{x}}_{\mathcal{B}}^{*}$ should be not parallel to $\vec{\mathbf{z}}_{\mathcal{B}}^{*}$.

The rotation matrix for the desired attitude is

$$\mathbf{R}^{*} = \begin{bmatrix} \vec{\mathbf{y}}_{\mathcal{B}}^{*} \times \vec{\mathbf{z}}_{\mathcal{B}}^{*} & \vec{\mathbf{y}}_{\mathcal{B}}^{*} & \vec{\mathbf{z}}_{\mathcal{B}}^{*} \end{bmatrix} \in \mathsf{SO}(3). \tag{15}$$

Therefore, the attitude error is as follows:

$$\mathbf{e}_{\mathbf{R}} = \frac{1}{2}\left[\mathbf{R}^{*T}\mathbf{R} - \mathbf{R}^{T}\mathbf{R}^{*}\right]^{\vee}, \tag{16}$$

where $^{\vee}$ is the vee map. The tracking error for the angular velocity is as follows:

$$\mathbf{e}_{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\mathcal{B}} - \mathbf{R}^{T}\mathbf{R}^{*}\left[\mathbf{R}^{*T}\dot{\mathbf{R}}^{*}\right]^{\vee}. \tag{17}$$

Finally, the control inputs (1) are chosen as follows:

$$\begin{cases} T = \left(k_{\mathbf{v}}\mathbf{e}_{\mathbf{v}} + mg\mathbf{e}_3\right)^{T}\mathbf{R}\mathbf{e}_3 \\ \boldsymbol{\tau} = -k_{\mathbf{R}}\mathbf{e}_{\mathbf{R}} - k_{\boldsymbol{\omega}}\mathbf{e}_{\boldsymbol{\omega}} + \left[\boldsymbol{\omega}_{\mathcal{B}}\right]^{\wedge}\mathbf{I}\boldsymbol{\omega}_{\mathcal{B}}, \end{cases} \tag{18}$$

where $^{\wedge}$ is the hat map, $k_{\mathbf{v}}$, $k_{\mathbf{R}}$ and $k_{\boldsymbol{\omega}}$ are some positive constant gains. The UAV receives this control input and converts it to the motor velocities.

4. Adaptive Fuzzy-Neuro Control Approach

4.1. Control Scheme and the Adaptive Fuzzy-Neuro Inference System

In the proposed control scheme which is shown in Fig. 3, the conventional PD controller is operating in parallel with the fuzzy-neuro controller (FNN block on Fig. 3). The conventional PD controller is utilized as an ordinary feedback controller to ensure the global asymptotic stability of the system in a compact space and provide sufficient time for the initialization of the learning process of the FNN. In this way, after a finite time, the FNN is supposed to learn the system dynamics and take over the control responsibility of the system. The PD control law is written in the following form:

$$\tau_c = k_p e + k_d \dot{e}. \tag{19}$$

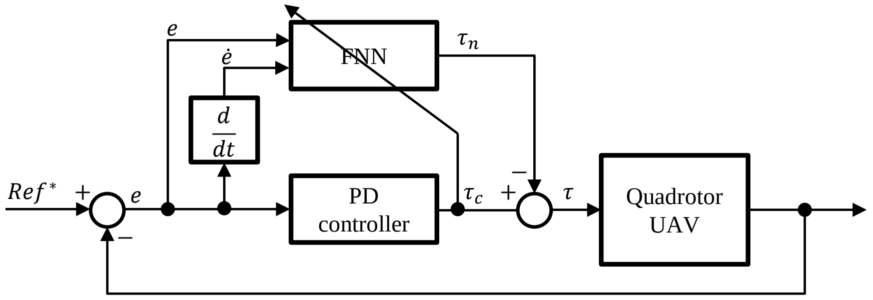


Figure 3: A conventional controller working in parallel with an intelligent controller.

where e is the feedback error, k_p and k_d are some positive constants corresponding to proportional and derivative gains respectively.

In this paper, the proposed FNN which has two input signals, $x_1(t) = e(t)$ and $x_2(t) = \dot{e}(t)$, and one output signal, $\tau_f(t)$, as depicted on Fig. 4 uses Takagi –Sugeno– Kang (TSK) fuzzy model in which the antecedent part is the fuzzy subset and the consequent part is the function of input variables. Hence, the k^{th} rule of TSK fuzzy model, where $k = (j-1).J + i$, with two input variables, x_1 and x_2, can be described as follows:

k^{th} Rule: *IF* x_1 is M_{1i} and x_2 is M_{2j}, *THEN* $f_{ij}{=}a_i x_1{+}b_j x_2{+}d_{ij}$

where a_i, b_j and d_{ij} are given constants ($i = 1, ..., I$ and $j = 1, ..., J$), M_{1i} and M_{2j} are fuzzy sets for the first and second input with their corresponding Gaussian membership functions represented as $\mu_{1_i}(x_1)$ and $\mu_{2_j}(x_2)$ respectively, and f_{ij} is the consequent part of the fuzzy system with I and J being the number of membership functions for x_1 and x_2 respectively. In the current investigation the coefficients a_i and b_j in the k^{th} rule of TSK fuzzy model are assumed to be equal to zero which is a widely used simplification which results in a zero-order TSK FNN model.

The strength of the k^{th} rule is calculated as the T-norm of the membership functions in the antecedent part. In our case, the T-norm is selected as

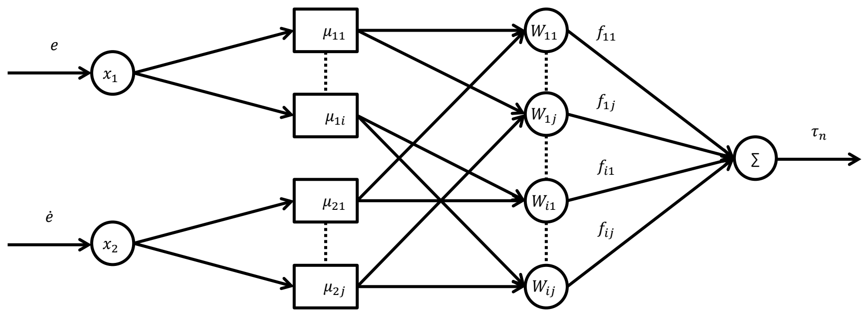


Figure 4: Structure of FNN.

multiplication to guarantee that the mathematical model is derivable which is required by the most neural network learning algorithms. Hence, the firing strength of the k^{th} rule can be written as follows:

$$W_{ij} = \mu_{1_i}(x_1)\mu_{2_j}(x_2) \tag{20}$$

where the Gaussian membership functions $\mu_{1_i}(x_1)$ and $\mu_{2_j}(x_2)$ in the above expression are defined as:

$$\mu_{1_i}(x_1) = \exp\left[-\frac{(x_1 - c_{1_i})^2}{\sigma_{1_i}^2}\right] \tag{21}$$

$$\mu_{2_j}(x_2) = \exp\left[-\frac{\left(x_2 - c_{2_j}\right)^2}{\sigma_{2_j}^2}\right] \tag{22}$$

where σ and c are the standard deviation and the mean of the membership functions, respectively. The real constants $c, \sigma > 0$ are among the tunable parameters of the FNN structure.

By inserting (21) and (22) into (20), we can obtain the following expression:

$$W_{ij} = \exp\left[-\frac{(x_1 - c_{1_i})^2}{\sigma_{1_i}^2} - \frac{\left(x_2 - c_{2_j}\right)^2}{\sigma_{2_j}^2}\right] \tag{23}$$

Therefore, the output signal of the FNN $\tau_n(t)$ can be computed as the weighted average of each rule's output:

$$\tau_n(t) = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} f_{ij}W_{ij}}{\sum_{i=1}^{I}\sum_{j=1}^{J} W_{ij}} = \sum_{i=1}^{I}\sum_{j=1}^{J} f_{ij}\overline{W}_{ij} \tag{24}$$

where $\overline{W}_{ij}$ is the normalized value of the output of the neuron ij from the second hidden layer of the network:

$$\overline{W}_{ij} = \frac{W_{ij}}{\sum_{i=1}^{I}\sum_{j=1}^{J} W_{ij}} \tag{25}$$

The overall control input τ to the system to be controlled is determined as follows:

$$\tau = \tau_c - \tau_n \tag{26}$$

where τ_n and τ_c are the control signals produced by the FNN controller and the PD controller, respectively.

In this investigation, the following assumptions have been used:

Based on the control cheme in Fig. 3, where the PD controller is used to ensure global asymptotic stability in compact space, it is assumed that the incoming signals, $x_1(t)$ and $x_2(t)$, and their time derivatives, $\dot{x}_1(t)$ and $\dot{x}_2(t)$, cannot have infinite values. Thus, they can be considered bounded:

$$|x_1(t)| \leq B_x, \;\; |x_2(t)| \leq B_x, \;\; |\dot{x}_1(t)| \leq B_{\dot{x}}, \;\; |\dot{x}_2(t)| \leq B_{\dot{x}} \quad \forall t \tag{27}$$

where the real constants B_x, $B_{\dot{x}} > 0$ are assumed to be some known numbers.

Similarly, the vectors defining the tuning parameters σ and c of the Gaussian membership functions are considered bounded as well:

$$\|\sigma_1\| \leq B_\sigma, \;\; \|\sigma_2\| \leq B_\sigma, \;\; \|c_1\| \leq B_c, \;\; \|c_2\| \leq B_c \tag{28}$$

where $\sigma_1 = [\sigma_{1_1} \ldots \sigma_{1_i} \ldots \sigma_{1_I}]^T$, $\sigma_2 = \left[\sigma_{2_1} \ldots \sigma_{2_j} \ldots \sigma_{2_J}\right]^T$, $c_1 = [c_{1_1} \ldots c_{1_i} \ldots c_{1_I}]^T$ and $c_2 = \left[c_{2_1} \ldots c_{2_j} \ldots c_{2_J}\right]^T$, and B_σ, $B_c > 0$ are some known constants.

Due to physical constraints, the time variable weight coefficient, $f_{ij}(t)$, can be considered bounded too, *i.e.*,

$$|f_{ij}(t)| \leq B_f \;\; \forall t \tag{29}$$

for some real constant $B_f > 0$.

From (20)-(25) and (27)-(28) it can be easily seen that $0 < \overline{W}_{ij} < 1$, and from (25) it is obvious that $\sum_{i=1}^{I} \sum_{j=1}^{J} \overline{W}_{ij} = 1$.

In addition, from (27) to (29) it is evident that τ and $\dot{\tau}$ will also be bounded signals:

$$|\tau\,(t)| \leq B_\tau, \;\; |\dot{\tau}\,(t)| \leq B_{\dot{\tau}} \quad \forall t \tag{30}$$

where $B_\tau, B_{\dot{\tau}} > 0$ are some known constants.

4.2. Sliding Mode Control Theory-Based Learning Algorithm for FNN

The zero dynamics of the learning error coordinate $\tau_c\,(t)$ can be described as a time-varying sliding surface S_c by utilizing the principles of the SMC theory:

$$S_c\,(\tau_n, \tau) = \tau_c\,(t) = \tau_n\,(t) + \tau\,(t) = 0 \tag{31}$$

By using this condition, the FNN structure is trained to become the nonlinear regulator which assists the conventional parallel controller (in our case PD controller) so that desired response can be obtained. Hence, the sliding surface for the nonlinear system under control is as follows:

$$S_p\,(e, \dot{e}) = \dot{e} + \lambda e \tag{32}$$

with $\lambda > 0$ being a parameter determining the reference trajectory of the error signal.

Definition: A sliding motion will appear on the sliding manifold $S_c(\tau_n, \tau) = \tau_c(t) = 0$ after a time t_h, if the condition $S_c(t)\dot{S}_c(t) = \tau_c(t)\,\dot{\tau}_c(t) < 0$ is satisfied for all t in some nontrivial semi-open subinterval of time of the form $[t, t_h) \subset (-\infty, t_h)$.

Since it is desired to design a dynamical feedback adaptation mechanism, or online learning algorithm for the FNN parameters such that the sliding mode condition of the above definition is enforced.

The adaptation theorem that summarizes the parameter update rules for FNN with two inputs is presented in the following form.

Theorem 1: If the adaptation laws for the parameters of the considered FNN are given as follows:

$$\dot{c}_{1i} = \dot{x_1} \tag{33}$$

$$\dot{c}_{2j} = \dot{x_2} \tag{34}$$

$$\dot{\sigma_{1i}} = -\frac{(\sigma_{1i})^3}{(x_1 - c_{1i})^2}\alpha sgn(\tau_c) \tag{35}$$

$$\dot{\sigma_{2j}} = -\frac{(\sigma_{2j})^3}{(x_2 - c_{2j})^2}\alpha sgn(\tau_c) \tag{36}$$

$$\dot{f}_{ij} = -\frac{\overline{W}_{ij}}{\overline{W}^T\overline{W}}\alpha \text{sign}\left(\tau_c\right) \tag{37}$$

$$\dot{\alpha} = \gamma|\tau_c| - \gamma\nu\alpha \tag{38}$$

where $\alpha > 0$ is the adaptive learning rate.

Then, the learning error $\tau_c(t)$ will converge to a small neighborhood of zero during a finite time t_h for any arbitrary initial condition $\tau_c(0)$.

Proof of Theorem 1: The time derivatives of (21) and (22) are as follows:

$$\dot{\mu_{1i}}(x_1) = -2A_{1i}(A_{1i})'\mu_{1i}(x_1) \tag{39}$$

$$\dot{\mu_{2j}}(x_2) = -2A_{2j}(A_{2j})'\mu_{2j}(x_2) \tag{40}$$

where

$$A_{1i} = \left(\frac{x_1 - c_{1i}}{\sigma_{1i}}\right) \quad and \quad A_{2j} = \left(\frac{x_2 - c_{2j}}{\sigma_{2j}}\right) \tag{41}$$

The time derivative of (25) can be obtained easily as follows:

$$\dot{\overline{W}}_{ij} = -\overline{W}_{ij}\dot{K}_{ij} + \overline{W}_{ij}\sum_{i=1}^{I}\sum_{j=1}^{J}\left(\overline{W}_{ij}\dot{K}_{ij}\right) \tag{42}$$

where

$$\dot{K}_{ij} = 2\Big(A_{1i}(A_{1i})' + A_{2j}(A_{2j})'\Big)$$

By using the following Lyapunov function, the stability condition can be checked:

$$V_c = \frac{1}{2}\tau_c^2(t) + \frac{1}{2\gamma}(\alpha - \alpha^*)^2 \tag{43}$$

The time derivative of V_c is given by:

$$\dot{V_c} \;=\; \tau_c\dot{\tau}_c = \tau_c(\dot{\tau}_n + \dot{\tau}) + \frac{1}{\gamma}\dot{\alpha}(\alpha - \alpha^*) \tag{44}$$

where

$$\dot{\tau_n} = \sum_{i=1}^{I}\sum_{j=1}^{J}(\dot{f_{ij}}\overline{W_{ij}} + f_{ij}\dot{\overline{W_{ij}}}) \tag{45}$$

By replacing (45) to the (44), the following equation is obtained:

$$\dot{V_c} = \tau_c\left(\sum_{i=1}^{I}\sum_{j=1}^{J}\Big(\dot{f_{ij}}\overline{W}_{ij} + f_{ij}\Big(-\overline{W}_{ij}\dot{K}_{ij} + \overline{W}_{ij}\sum_{i=1}^{I}\sum_{j=1}^{J}\overline{W}_{ij}\dot{K}_{ij}\Big)\Big) + \dot{\tau}\right)$$
$$+ \frac{1}{\gamma}\dot{\alpha}(\alpha - \alpha^*)$$

$$\dot{V_c} = \tau_c\Bigg[\sum_{i=1}^{I}\sum_{j=1}^{J}\dot{f_{ij}}\overline{W}_{ij} - 2\sum_{i=1}^{I}\sum_{j=1}^{J}\overline{W}_{ij}\Big(A_{1i}(A_{1i})' + A_{2j}(A_{2j})'\Big)f_{ij}$$
$$+ 2\sum_{i=1}^{I}\sum_{j=1}^{J}\Big(\overline{W}_{ij}f_{ij}\sum_{i=1}^{I}\sum_{j=1}^{J}\overline{W}_{ij}\Big(A_{1i}(A_{1i})' + A_{2j}(A_{2j})'\Big)\Big) + \dot{\tau}\Bigg]$$
$$+ \frac{1}{\gamma}\dot{\alpha}(\alpha - \alpha^*),$$

where

$$\dot{A_{1i}} = \frac{(\dot{x_1} - \dot{c_{1i}})\sigma_{1i} - (x_1 - c_{1i})\dot{\sigma_{1i}}}{{\sigma_{1i}}^2}$$

$$\dot{A_{2j}} = \frac{(\dot{x_2} - \dot{c_{2j}})\sigma_{2j} - (x_2 - c_{2j})\dot{\sigma_{2j}}}{{\sigma_{2j}}^2}$$

(46) can be obtained by using (33)-(36);

$$A_{1i}\dot{A_{1i}} = A_{2j}\dot{A_{2j}} = \alpha sgn(\tau_c) \tag{46}$$

$$\dot{V}_c = \tau_c\Big[\sum_{i=1}^{I}\sum_{j=1}^{J}\dot{f}_{ij}\overline{W}_{ij} - 4\sum_{i=1}^{I}\sum_{j=1}^{J}\overline{W}_{ij}\big(\alpha sgn(\tau_c)\big)f_{ij} +$$
$$+ 4\sum_{i=1}^{I}\sum_{j=1}^{J}\Big(\overline{W}_{ij}f_{ij}\sum_{i=1}^{I}\sum_{j=1}^{J}\overline{W}_{ij}(\alpha sgn\big(\tau_c)\big)\Big) + \dot{\tau}\Big] + \frac{1}{\gamma}\dot{\alpha}(\alpha - \alpha^*)$$

Since $\sum_{i=1}^{I}\sum_{j=1}^{J}\widetilde{\overline{W_{ij}}} = 1$,

$$\dot{V}_c = \tau_c\Big[\sum_{i=1}^{I}\sum_{j=1}^{J}\dot{f}_{ij}\overline{W}_{ij} + \dot{\tau}\Big] + \frac{1}{\gamma}\dot{\alpha}(\alpha - \alpha^*),$$

where

$$\dot{f}_{ij} = -\frac{\overline{W}_{ij}}{\overline{W}^T\overline{W}}\alpha \text{sign}\left(\tau_c\right) \tag{47}$$

So that the following equation for the time derivative of Lyapunov function is achieved.

$$\dot{V}_c = \tau_c\Big[-\alpha sgn(\tau_c) + \dot{\tau}\Big] + \frac{1}{\gamma}\dot{\alpha}(\alpha - \alpha^*) \tag{48}$$

Furthermore,

$$\begin{aligned} \dot{V}_c &= -\alpha|\tau_c| + |\tau_c|B_{\dot{\tau}} + \frac{1}{\gamma}\dot{\alpha}(\alpha - \alpha^*) && (49) \\ &= -(\alpha - \alpha^*)|\tau_c| - \alpha^*|\tau_c| + |\tau_c|B_{\dot{\tau}} + \frac{1}{\gamma}\dot{\alpha}(\alpha - \alpha^*) && (50) \end{aligned}$$

Considering the adaptation law of α as follows,

$$\dot{\alpha} = \gamma|\tau_c| - \gamma\nu\alpha \tag{51}$$

we have:

$$\dot{V}_c = |\tau_c|B_{\dot{\tau}} - \alpha^*|\tau_c| - \nu(\alpha - \frac{1}{2}\alpha^*)^2 + \frac{\nu}{4}\alpha^{*2} \tag{52}$$

Taking α^* as $B_{\dot{\tau}} \leq \frac{\alpha^*}{2}$, we have:

$$\dot{V}_c \leq -\frac{\alpha^*}{2}|\tau_c| + \frac{\nu}{4}\alpha^{*2} \tag{53}$$

which implies that the Lyapunov function decreases until $|\tau_c| < \frac{\nu\alpha^*}{2}$. So that τ_c will stay bounded. Furthermore ν is a design parameter and it is possible to take this value as small as desired.

The relation between the sliding function (it is a point in this investigation) S_p and the zero adaptive learning error level S_c is as follows:

$$S_c = \tau_c = k_P e = k_p S_p \tag{54}$$

The tracking performance of the feedback control system can be analyzed by introducing the following Lyapunov function candidate:

$$V_p = \frac{1}{2} S_p^2 \tag{55}$$

Theorem 2: If the adaptation law for the adjustable parameters of the FNN is selected as in (33)-(38), then the negative definiteness of the time derivative of the Lyapunov function in (55) is guaranteed.

Proof of Theorem 2: Evaluating the time derivative of the Lyapunov function in (55) yields:

$$\dot{V}_c \leq -\frac{\alpha^*}{2}|\tau_c| + \frac{\nu}{4}\alpha^{*2} \qquad \forall S_c, S_p \neq 0 \tag{56}$$

This equation implies that V_c converges until $|\tau_c| < \frac{\nu\alpha^*}{2}$ and $|\tau_c|$ remains bounded.

4.3. Levenberg-Marquardt-Based Training Algorithm for FNN

Theorem 3: If the Levenberg-Marquardt based parameter update rules for FNNs are as follows:

$$\dot{c}_{1i} = \dot{x}_1 \tag{57}$$

$$\dot{c}_{2j} = \dot{x}_2 \tag{58}$$

$$\dot{\sigma_{1i}} = -\frac{(\sigma_{1i})^3}{(x_1 - c_{1i})^2}\alpha sgn(\tau_c) \tag{59}$$

$$\dot{\sigma_{2j}} = -\frac{(\sigma_{2j})^3}{(x_2 - c_{2j})^2}\alpha sgn(\tau_c) \tag{60}$$

$$\dot{f} = -\gamma\left(\overline{W}\,\overline{W}^T + \delta I\right)^{-1}\overline{W} sgn(\tau_c) \tag{61}$$

where δ is the adaptive parameter considered for the Levenberg Marquardt and is selected as equal to

$$\delta = max\{\overline{W}^T\overline{W}, \overline{\alpha}\} \tag{62}$$

In which $\overline{\alpha}$ has a constant value.

Then, the learning error $\tau_c(t)$ will converge to a small neighborhood of zero during a finite time t_h for any arbitrary initial condition $\tau_c(0)$.

Proof of Theorem 3: In order to prove the stability of Theorem 3, the Lyapunov function considered is as follows:

$$V_c = \frac{1}{2}\tau_c^2(t) \tag{63}$$

Using the adaptation laws of (57)-(60) and a similar analysis as in Appendix B we have:

$$\dot{V}_c = \tau_c\Big[\sum_{i=1}^{I}\sum_{j=1}^{J}\dot{f}_{ij}\overline{W}_{ij} + \dot{\tau}\Big] \tag{64}$$

Considering the adaptation law of (61), the time derivative of the Lyapunov function can be rewritten as follows:

$$\begin{aligned}
\dot{V}_c &= \tau_c\Big[\gamma\overline{W}^T\Big(-\delta^{-1} + \delta^{-1}\overline{W}\big(I + \overline{W}^T\delta^{-1}\overline{W}\big)^{-1}\overline{W}^T\delta^{-1}\Big)\overline{W}sgn(\tau_c) + \dot{\tau}\Big] \\
&= \tau_c\Big[\gamma\overline{W}^T\Big(-\delta^{-1} + \delta^{-1}\overline{W}\big(\delta + \overline{W}^T\overline{W}\big)^{-1}\overline{W}^T\Big)\overline{W}sgn(\tau_c) + \dot{\tau}\Big] \\
&= -\delta^{-1}\gamma\overline{W}^T\overline{W}|\tau_c| + |\tau_c|\delta^{-1}\gamma\overline{W}^T\overline{W}\left(\delta + \overline{W}^T\overline{W}\right)^{-1}\overline{W}^T\overline{W} + B_{\dot{\tau}}|\tau_c| \\
&= -\delta^{-1}\gamma\overline{W}^T\overline{W}|\tau_c| + |\tau_c|\delta^{-1}\gamma\overline{W}^T\overline{W}\left(\delta + \overline{W}^T\overline{W}\right)^{-1}\left(\delta + \overline{W}^T\overline{W} - \delta\right) \\
&+ B_{\dot{\tau}}|\tau_c| \\
&= -\delta^{-1}\gamma\overline{W}^T\overline{W}|\tau_c| + \gamma|\tau_c|\delta^{-1}\overline{W}^T\overline{W} - \gamma|\tau_c|\overline{W}^T\overline{W}\left(\delta + \overline{W}^T\overline{W}\right)^{-1} + B_{\dot{\tau}}|\tau_c| \\
&= -\gamma|\tau_c|\overline{W}^T\overline{W}\left(\delta + \overline{W}^T\overline{W}\right)^{-1} + B_{\dot{\tau}}|\tau_c|.
\end{aligned}$$

Considering δ as equal to $\delta = \overline{W}^T\overline{W}$, we have:

$$\dot{V}_c \;=\; -0.5\gamma|\tau_c| + B_{\dot{\tau}}|\tau_c| \tag{65}$$

It is further assumed that $\gamma > \frac{B_{\dot{\tau}}}{4}$, so that:

$$\dot{V}_c \;\leq\; -\frac{1}{4}\gamma|\tau_c| \tag{66}$$

This concludes the proof.

In order to analyze the sliding behavior of the system under Levenberg-Marquardt adaptation laws of 57-61, the following Lyapunov function candidate is used.

$$V_p = \frac{1}{2}S_p^2 \tag{67}$$

Theorem 4: Let the adjustable parameters of FNN be tuned using the adaptation laws as in (57)-(61), the negative definiteness of the time derivative of the Lyapunov function in (67) is guaranteed and the sliding manifold of $S_p = \dot{e} + \chi e$ in which $\chi = K_p/K_d$ converges to *zero*.

Proof of Theorem 4: The following equation relates the sliding manifold of $S_p = \dot{e} + \chi e$ to τ_c.

$$S_p = \dot{e} + \chi e = \dot{e} + \frac{K_p}{Kd} e = \frac{\tau_c}{K_d} \tag{68}$$

Hence, the time derivative of the Lyapunov function in (67) yields:

$$\dot{V}_c = S_p \dot{S}_p = \frac{\tau_c \dot{\tau}_c}{K_d^2} \leq \frac{-\gamma |\tau_c|}{4K_d^2} = -\frac{\gamma |S_p|}{4K_d} \tag{69}$$

which implies that V_c converges to zero in finite time t_h which is as follows.

$$t_h \leq \frac{4K_d S_p(0)}{\gamma} \tag{70}$$

5. Simulation Studies

In this section, the trajectory-tracking task in which a quadrotor aerial vehicle which needs to convergence to a desired path in finite time is performed under wind and gust conditions. The simulation results are presented to illustrate the performance of the proposed controller.

5.1. Intrinsic Parameters of the Quadrotor and Control Variables

Robot operating system (ROS) and Gazebo simulator are used to implement the dynamical simulations of trajectory-tracking task of quadrotor where its intrinsic parameters are defined in Table 1. These parameters are selected to be close to the ones of the real Parrot Ar.Drone 2.0 quadrotor. The control gains for the PD controller are chosen as follows:

$$k_p = 2.5, \quad k_d = 0.005$$

Table 1: Quadrotor's intrinsic parameters.

Parameter	**Value**	**Unit**
m	1.07	[kg]
l	0.17	[m]
b	6.55×10^{-6}	$[\mathrm{N} \cdot \mathrm{s}^2]$
d	2.66×10^{-2}	$[\mathrm{N} \cdot \mathrm{m} \cdot \mathrm{s}^2]$
I_x	3.0×10^{-2}	$[\mathrm{kg} \cdot \mathrm{m}^2]$
I_y	4.0×10^{-2}	$[\mathrm{kg} \cdot \mathrm{m}^2]$
I_z	2.1×10^{-2}	$[\mathrm{kg} \cdot \mathrm{m}^2]$

As for the initial control parameters of the FNN controller, it is set to (here, subindexes x, y and z correspond to position channels in Fig. 2):

$$c_{1_x} = c_{1_y} = c_{1_z} = [-3,\ 0.0001,\ 3]^T$$

$$c_{2_x} = c_{2_y} = c_{2_z} = [-30,\ 0.0001,\ 30]^T$$

$$\sigma_{1_x} = \sigma_{1_y} = \sigma_{1_z} = [1.5,\ 1.5,\ 1.5]^T$$

$$\sigma_{2_x} = \sigma_{2_y} = \sigma_{2_z} = [15,\ 15,\ 15]^T$$

$$\alpha_x = \alpha_y = \alpha_z = 0.001$$

while the initial condition of time variable weight coefficient, $f_{ij}(0)$, is chosen to be sufficiently small, *i.e.*, $f_{ij}(0) \in [0, 0.001]$.

The adaptive learning parameters for FNN are chosen as:

- FNN based on SMC:
$$\gamma_x = \gamma_y = \gamma_z = 1.5,$$
$$\nu_x = 0.85,\ \nu_y = 0.85,\ \nu_z = 0.2$$

- FNN based on LM:
$$\gamma_x = \gamma_y = \gamma_z = 1.5,$$
$$\bar{\alpha_x} = 1.1,\ \bar{\alpha_y} = 1.1,\ \bar{\alpha_z} = 0.9$$

which are updated on intervals of $dt = 0.01$s, while the total simulations time is equal to 50s.

5.2. Trajectory Generation

In the classical UAV flight missions, climbing, ascending and descending curves as well as level flight are considered as typical flight manoeuvres. Therefore, in our experimental scenario, all aforementioned manoeuvres will be included during the flight of quadrotor in order to evaluate the robustness of proposed controllers under the flight sequence which resembles the actual UAV flights. At the beginning, the quadrotor UAV is hovering at 15m height (an initial position). Then, it starts to make two full circles with 20m diameters in clockwise followed by counterclockwise direction. At the same time, the quadrotor is also changing its altitude before coming back to its initial position. Throughout the simulation, the desired translational velocity are kept constant and equal to 2.5m/s.

In addition, the feasibility of flight under the dynamic constraints of quadrotor are ensured by saturating the control input signals and defining trajectory as a time-based trajectory.

5.3. Source of Disturbances

In order to evaluate the efficacy of proposed control strategy, the periodic wind gust is added to our simulation scenario. As can be seen from Fig. 5, the wind gust v_w blows with three different speed (1.0m/s, 3.0m/s and 5.0m/s) along the $[-1, -1, 0]$ direction for the first 17s, then its orientation changes to opposite direction to change back to original direction after 34s.

5.4. Simulation Results

In Fig. 6, the trajectory tracking in the presence of wind (v_w = 3m/s) is shown for the PD controller and two FNN controllers which are tuned by SMC and LM approach, respectively, operating in parallel with the conventional PD controller. From these figures, it is evident that the PD controller has a notable steady state error occurred from internal uncertainties such as lack of modeling as well as due to the external disturbance as the periodic wind. In addition, when the PD controller works alone, it cannot eliminate the existed error during the whole simulation. However, in case of SMC-based FNN (SMCFNN) and LM-based FNN (LMFNN) controllers, the steady state error is notably reduced because of adaptive learning capabilities of FNN structure. As a result, trajectory tracking performance of quadrotor which uses intelligent FNN structure becomes significantly better compared to normal case when the PD controller is only utilized.

As for output control signals, Figs. 7-8 present control signals for x, y and z axes in case when the PD controller is operating in parallel with FNN controller which are tuned by SMC and LM approach, respectively. As can be seen from these figures, the FNN controller is taking over the control responsibilities from PD controller, and therefore, after some time the output control signal from PD controller approaches to zero neighborhood, and then only FNN controls the system as it is supposed to do in such kind of control schemes. It should be noted that when trajectory sequence changes or some disturbances occur

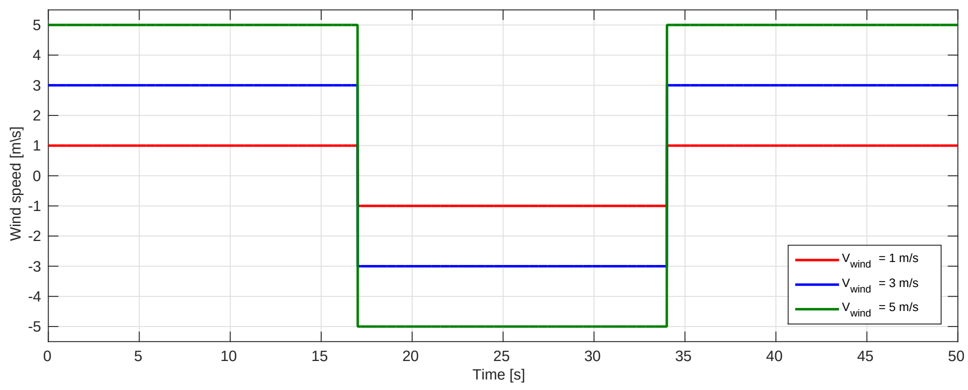
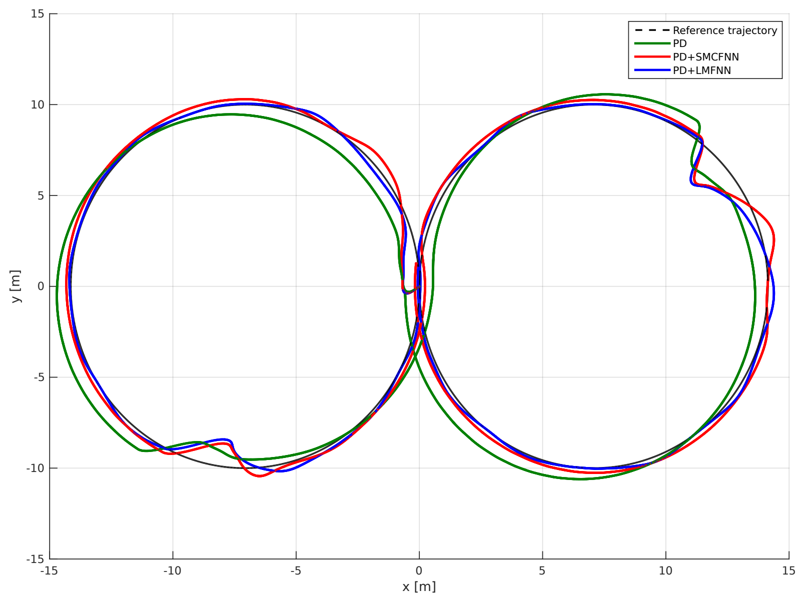
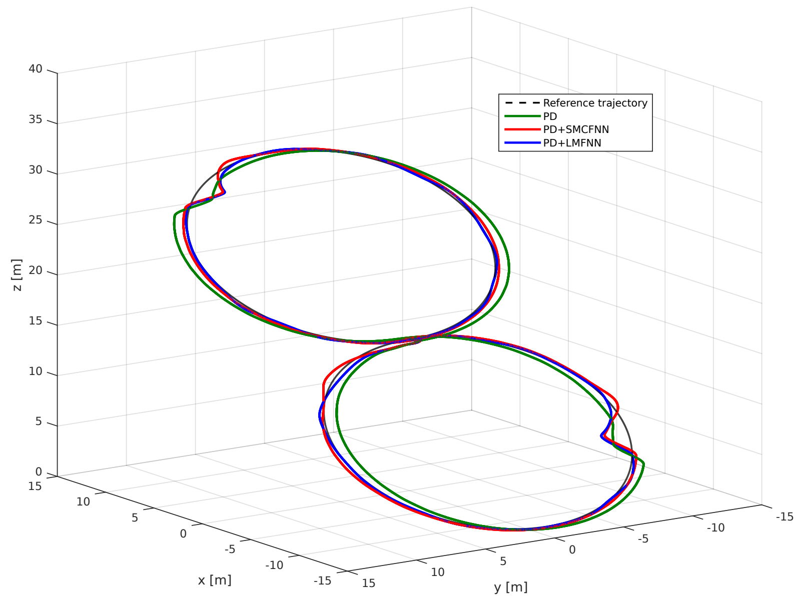


Figure 5: Wind profile



(a) in xy-plane

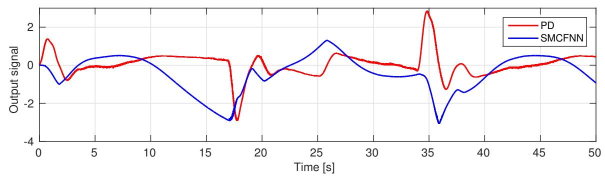


(b) in 3D view

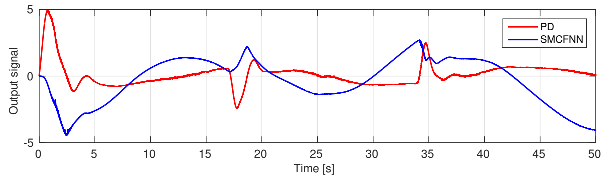
Figure 6: Trajectory tracking for $v_w = 3$m/s.

the output control signals from the PD controller become nonzero. In such case, FNN restarts the learning process and takes over control responsibilities again as shown in Figs. 7-8.

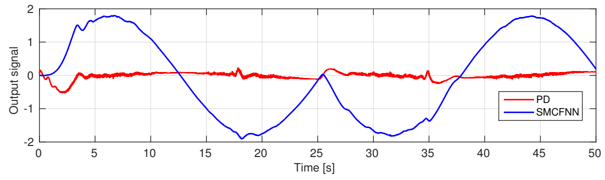
The Euclidean error in the presence of wind is shown in Fig. 9. The combination of PD and FNN controllers, PD+SMCFNN and PD+LMFNN, give a significantly smaller error than the conventional PD controller when it works



(a) in x-axis

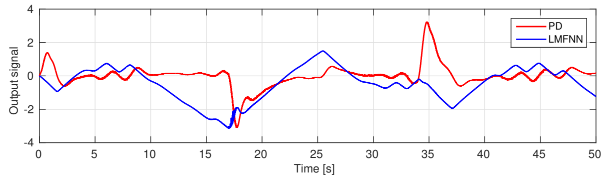


(b) in y-axis

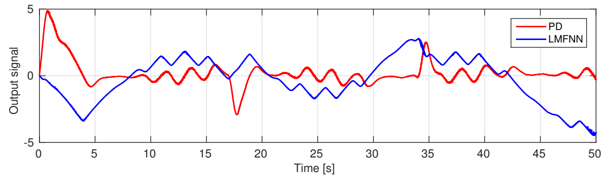


(c) in z-axis

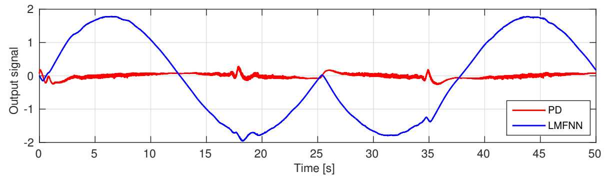
Figure 7: PD and SMCFNN control signals for the x, y and z axes for $v_w = 3\text{m/s}$.



(a) in x-axis



(b) in y-axis



(c) in z-axis

Figure 8: PD and LMFNN control signals for the x, y and z axes for $v_w = 3\text{m/s}$.

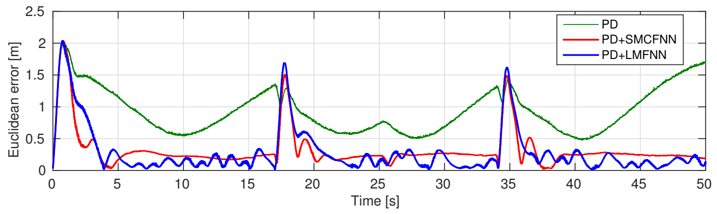


Figure 9: Euclidean error of the different controllers for $v_w = 3$m/s.

alone. This can be also seen from Table 2 which shows the average root mean squared error (RMSE) values for ten simulations for each case: 1.0m/s, 3.0m/s and 5.0m/s. It is observed that FNN controllers decrease the PD controller RMSE error for different wind speed by 67%, 55% and 50%, respectively. In addition, it can be also observed that LMFNN outperforms SMCFNN with lower wind gust. On the other hand, the PD controller can be tuned in more aggressive way to achieve superior results, although this is not practical in real life due to the lack of modelling and unknown disturbances in real-time applications. Furthermore, aggressive tuning tends to be case dependent, and therefore, it cannot give a comparable performance in different conditions; while adaptive learning capabilities of FNN structure are essential for real world applications.

6. Experimental Tests

The experimental flight tests for the trajectory tracking problem were conducted in the indoor environment and evaluated in the Motion Capture Laboratory at Nanyang Technological University (NTU), Singapore, shown in Fig. 10. The laboratory environment is designed to use a set of eight OptiTrack Prime 13 cameras, which are fed into an OptiHub, to provide real-time pose (position and attitude) measurements of the UAV's COM with an update rate of 120Hz and accuracy around 0.1mm. The OptiTrack cameras are able to recognize a particular UAV according to the pattern of the infrared reflective markers which are fixed on the UAV's frame. The pose data are routed to the controller trough the Optitrack server. The aircraft used for the experimental

Table 2: Average Euclidean RMSE for different wind speeds (unit: m).

Controller	**Wind speed**, v_w		
	1m/s	3m/s	5m/s
PD	0.966	1.006	1.204
SMC-based FNN	0.337	0.452	0.600
LM-based FNN	0.316	0.477	0.689

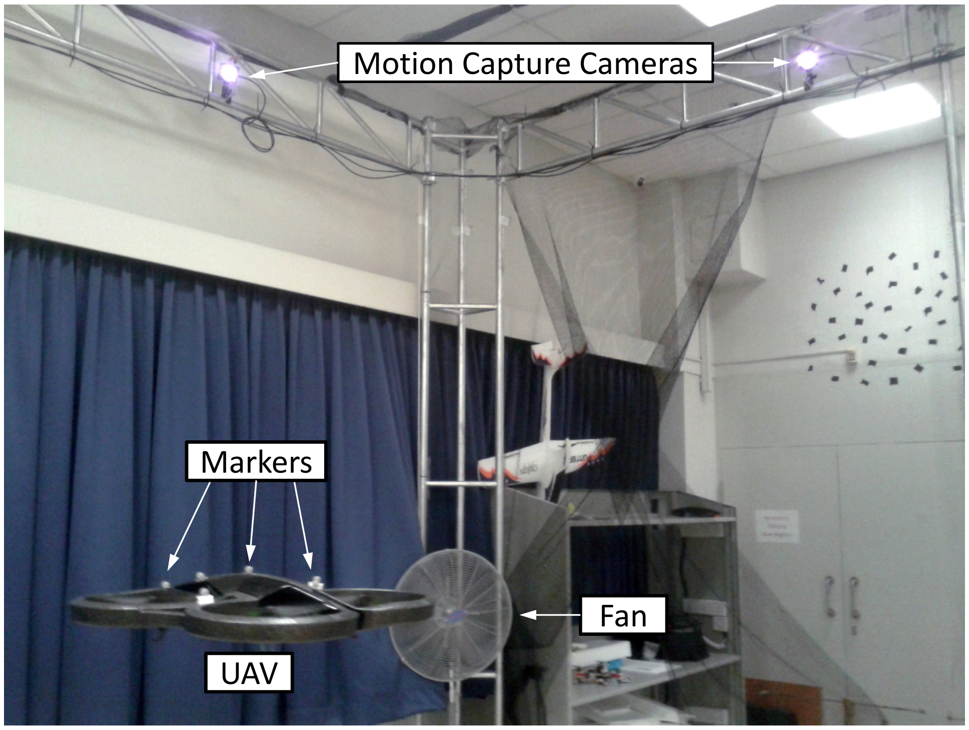


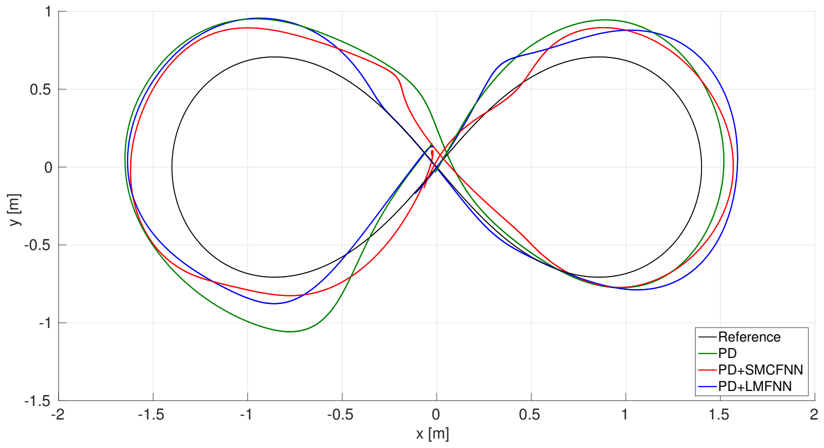
Figure 10: Indoor experimental setup.

flight tests is Parrot AR.Drone 2.0 Power Edition UAV, which is a velocity controlled commercial quadrotor. This UAV features a 1GHz CPU, 800MHz GPU, 1GB of RAM, two video cameras, inertial and ultrasonic sensors. In order to communicate with this UAV, it creates its own Wi-Fi network and opens an UDP communication to receive command signals, allowing to connect from devices with different operative systems such as Linux, Android or Windows. The control system itself is implemented in Linux using ROS and C++ environment. The Ardrone Autonomy ROS package is used to transfer the output velocity from the controller to the quadrotor via a Wi-Fi connection.

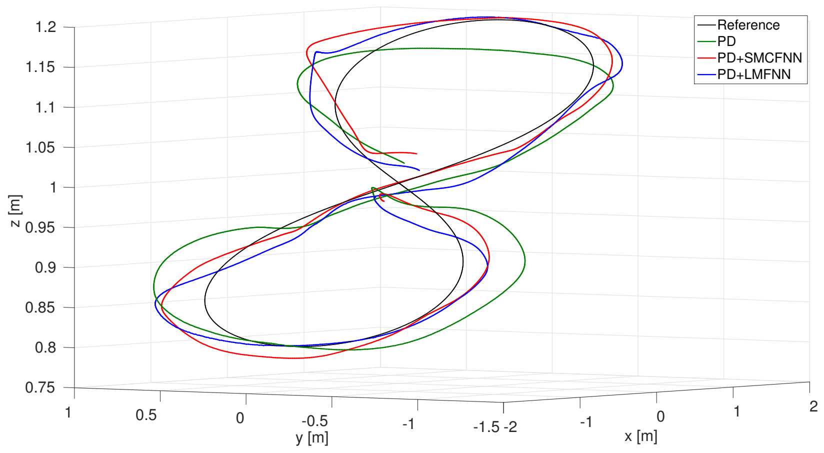
6.1. Experimental Results

In this section, the performance of PD controller only and FNN controller trained by SMC and LM working in parallel with the conventional PD controller for $'8'$ shaped time based trajectory is presented. The maximum speed along trajectory is kept to be 1m/s. In order to evaluate the efficacy of proposed control strategy, an industrial fan, which generates maximum wind gust of 2m/s, is used to imitate the external disturbances. The wind gust blows along the $[1, -1, 0]$ direction.

In Fig. 11, the trajectory tracking is shown for the PD controller and FNN controllers which are tuned by SMC and LM. As can be seen, the steady state error is notably reduced because of learning capabilities of FNN structure.



(a) in xy-plane



(b) in 3D view

Figure 11: Trajectory tracking for $v_w = 2$m/s.

This can also be seen from the Euclidean error and the average RMSE values from ten experiments which are shown in Fig. 12 and Table 3, respectively. The combination of PD and FNN controller gives a significantly less error than the conventional PD controller when it works alone. It should be noted that FNN controllers decrease the PD controller RMSE error by about 36%. Since we do not give any step input to the system, we do not provide the transient response characteristics of the system, such as rise time, peak time and maximum overshoot. A demonstration video related to our experiments can be found at: `https://youtu.be/yeUAsIHaZ20`.

Hence, the trajectory tracking performance of quadrotor which uses intelligent FNN structure becomes significantly better compared to normal case when the PD controller is only utilized. However, SMC and LM-based FNN controllers were not able completely to take over the control responsibilities

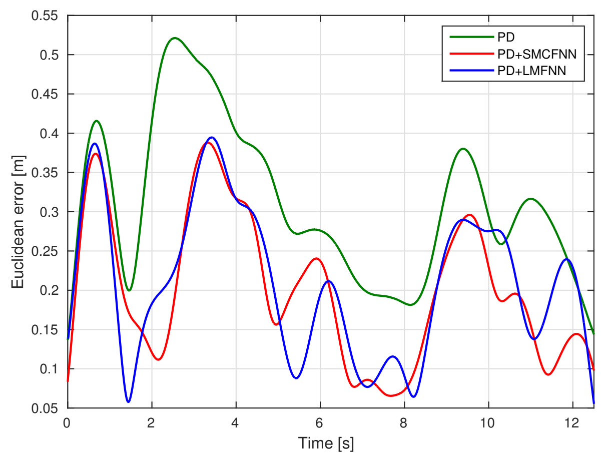
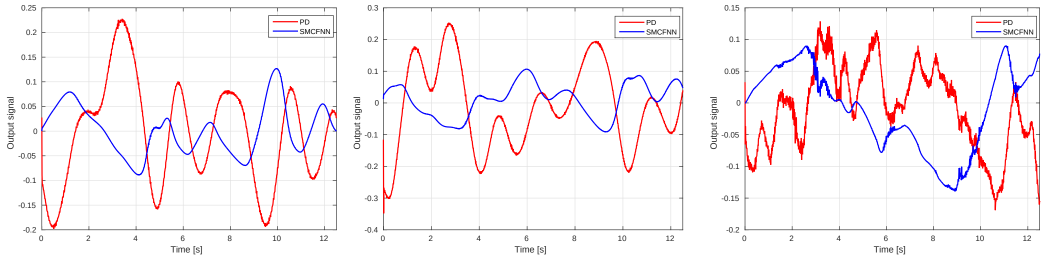


Figure 12: Euclidean error for $v_w = 2\text{m/s}$.



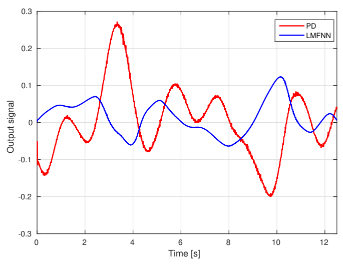
(a) in x-axis (b) in y-axis (c) in z-axis

Figure 13: SMCFNN and PD control signals for the x, y and z axes for $v_w = 2\text{m/s}$.

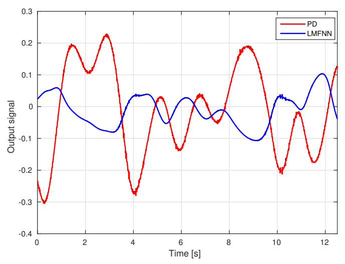
from PD controller as shown in Fig. 13 and 14. Whereas both FNN controllers were dominating in simulation studies, it was not the case for the real-time tests. The reason for this is the space limitation of our Motion Capture Lab which measures $5 \times 7\text{m}^2$. Therefore, the active utilized area for the UAV is maximum $3 \times 5\text{m}^2$. In such a small area, also by having challenging trajectory, the FNN controller does not have enough time to learn. In addition, there exists communication delay between computer and ArDrone UAV. This latency is mainly caused by the WiFi protocol delay as well as the down-sampling/buffering step performed by ArDrone firmware prior to sending the feedback over WiFi.

Table 3: Average Euclidean RMSE for different controllers (unit: m).

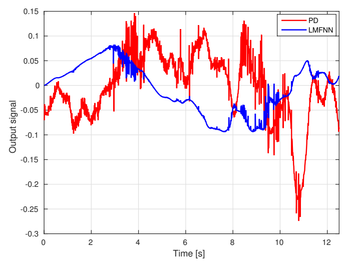
Controller	**PD**	**SMC-based FNN**	**LM-based FNN**
RMSE	0.327	0.210	0.228



(a) in x-axis



(b) in y-axis



(c) in z-axis

Figure 14: LMFNN and PD control signals for the x, y and z axes for $v_w = 2\text{m/s}$.

7. Conclusion

In this paper, SMC theory and LM-based learning algorithm for intelligent FNN controller are proposed for the control and stabilizing of underactuated nonlinear quadrotor UAV along a predefined trajectory in the presence of wind gust conditions. The stability analysis of proposed parameter update rules are presented. It was also demonstrated that proposed methods are capable to significantly reducing the steady state errors and overcome the disturbances and existed uncertainties which are generated by lack of modeling. Extensive simulations in ROS and Gazebo environment are conducted to evaluate the performance of the proposed controllers with the conventional PD controller. In order to further test the proposed methods, the real-time experiments have been also performed by using OptiTrack Motion Capture System. Experimental results show that the combination of PD and FNN which is tuned by SMC and LM algorithms gives a significantly less steady state error than the conventional PD controller when it works alone.

7.1. Future Work

Future research will include the implementation of state estimation methods to account for the available on board information. In addition, type-2 fuzzy sets will be implemented to achieve better noise rejection property. Moreover, we will conduct real-time experiments with different types of UAVs in order to validate the portability of our controller.

Acknowledgment

The research was partially supported by the ST Engineering - NTU Corporate Lab through the NRF corporate lab@university scheme.

References

Alejo, D., Cobano, J. A., Heredia, G., Ollero, A., 2014. Collision-free 4d trajectory planning in unmanned aerial vehicles for assembly and structure construction. Journal of Intelligent & Robotic Systems 73 (1-4), 783.

Castillo, O., Amador-Angulo, L., Castro, J. R., Garcia-Valdez, M., 2016a. A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems. Information Sciences 354, 257 – 274.

Castillo, O., Castro, J. R., Melin, P., Rodriguez-Diaz, A., 2013. Universal approximation of a class of interval type-2 fuzzy neural networks in nonlinear identification. Advances in Fuzzy Systems 2013, 7.

Castillo, O., Cervantes, L., Soria, J., Sanchez, M., Castro, J. R., 2016b. A generalized type-2 fuzzy granular approach with applications to aerospace. Information Sciences 354, 165 – 177.

Celikyilmaz, A., Turksen, I. B., 2009. Modeling Uncertainty with Fuzzy Logic: With Recent Theory and Applications, 1st Edition. Springer Publishing Company, Incorporated.

Cervantes, L., Castillo, O., 2015. Type-2 fuzzy logic aggregation of multiple fuzzy controllers for airplane flight control. Information Sciences 324, 247 – 256.

Dong, Y., Jun, F., Bin, Y., Youmin, Z., Jianliang, A., July 2015. Position and Heading Angle Control of an Unmanned Quadrotor Helicopter Using LQR Method. In: Control Conference (CCC), 2015 34th Chinese. pp. 5566–5571.

Eresen, A., Imamoglu, N., Efe, M. O., 2012. Autonomous Quadrotor Flight with Vision-Based Obstacle Avoidance in Virtual Environment. Expert Systems with Applications 39 (1), 894 – 905.

Fakurian, F., Menhaj, M. B., Mohammadi, A., Oct 2014. Design of a fuzzy controller by minimum controlling inputs for a quadrotor. In: Robotics and Mechatronics (ICRoM), 2014 Second RSI/ISM International Conference on. pp. 619–624.

Fu, C., Sarabakha, A., Kayacan, E., Wagner, C., John, R., Garibaldi, J. M., 2016. A comparative study on the control of quadcopter uavs by using singleton and non-singleton fuzzy logic controllers. In: 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1023–1030.

Gaxiola, F., Melin, P., Valdez, F., Castillo, O., 2014. Interval type-2 fuzzy weight adjustment for backpropagation neural networks with application in time series prediction. Information Sciences 260, 1 – 14.

Gaxiola, F., Melin, P., Valdez, F., Castillo, O., 2015. Generalized type-2 fuzzy weight adjustment for backpropagation neural networks in time series prediction. Information Sciences 325, 159 – 174.

Hagras, H. A., 2004. A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. IEEE Transactions on Fuzzy Systems 12 (4), 524–539.

Hernandez-Gonzalez, M., Alanis, A., Hernandez-Vargas, E., 2012. Decentralized Discrete-Time Neural Control for a Quanser 2-DOF Helicopter. Applied Soft Computing 12 (8), 2462 – 2469.

Hofer, M., Muehlebach, M., D'Andrea, R., 2016. Application of an approximate model predictive control scheme on an unmanned aerial vehicle. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 2952–2957.

Hua, M. D., Hamel, T., Morin, P., Samson, C., Feb 2013. Introduction to feedback control of underactuated vtol vehicles: A review of basic control design ideas and principles. IEEE Control Systems 33 (1), 61–75.

Khanesar, M. A., Kayacan, E., Aug 2015. Levenberg-Marquardt Training Method for Type-2 Fuzzy Neural Networks and its Stability Analysis. In: Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on. pp. 1–7.

Kim, C. J., Chwa, D., 2015. Obstacle avoidance method for wheeled mobile robots using interval type-2 fuzzy neural network. IEEE Transactions on Fuzzy Systems 23 (3), 677–687.

Kumbasar, T., Hagras, H., 2014. Big Bang-Big Crunch optimization based interval type-2 fuzzy PID cascade controller design strategy. Information Sciences 282, 277 – 295.

Lee, T., Leok, M., McClamroch, N. H., June 2012. Nonlinear robust tracking control of a quadrotor UAV on SE(3). In: 2012 American Control Conference (ACC). pp. 4649–4654.

Lin, F. J., Hung, Y. C., Ruan, K. C., 2014. An intelligent second-order sliding-mode control for an electric power steering system using a wavelet fuzzy neural network. IEEE Transactions on Fuzzy Systems 22 (6), 1598–1611.

Ma, X., Sun, F., Li, H., He, B., 2017. Neural-network-based sliding-mode control for multiple rigid-body attitude tracking with inertial information completely unknown. Information Sciences 400–401, 91 – 104.

Mahony, R., Kumar, V., Corke, P., 2012. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. IEEE Robotics Automation Magazine 19 (3), 20–32.

Martinez-Martinez, V., Gomez-Gil, F. J., Gomez-Gil, J., Ruiz-Gonzalez, R., 2015. An Artificial Neural Network Based Expert System Fitted with Genetic Algorithms for Detecting the Status of Several Rotary Components in Agro-Industrial Machines Using a Single Vibration Signal. Expert Systems with Applications 42 (17–18), 6433 – 6441.

Mendel, J., Hagras, H., Tan, W.-W., Melek, W. W., Ying, H., 2014. Introduction to type-2 fuzzy logic control: theory and applications. John Wiley & Sons.

Michael, N., Fink, J., Kumar, V., 2011. Cooperative manipulation and transportation with aerial robots. Autonomous Robots 30 (1), 73–86.

Mukherjee, I., Routroy, S., 2012. Comparing the Performance of Neural Networks Developed by Using Levenberg–Marquardt and Quasi-Newton with the Gradient Descent Algorithm for Modelling a Multiple Response Grinding Process. Expert Systems with Applications 39 (3), 2397 – 2407.

Olivares-Mendez, M. A., Fu, C., Ludivig, P., Bissyande, T. F., Kannan, S., Zurad, M., Annaiyan, A., Voos, H., Campoy, P., 2015. Towards an Autonomous Vision-Based Unmanned Aerial System against Wildlife Poachers. Sensors 15 (12), 29861.

Perez-Ortiz, M., Pena, J. M., Gutierrez, P. A., Torres-Sanchez, J., Hervas-Martinez, C., Lopez-Granados, F., 2016. Selecting Patterns and Features for Between- and Within- Crop-Row Weed Mapping Using UAV-Imagery. Expert Systems with Applications 47, 85 – 94.

Salim, J., Ismail, M., Suwarno, I., Nawi, N. M., Khan, A., Rehman, M., 2013. A New Levenberg Marquardt based Back Propagation Algorithm Trained with Cuckoo Search. 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013 11, 18 – 23.

Salimifard, M., Safavi, A. A., 2013. Nonlinear system identification based on a novel adaptive fuzzy wavelet neural network. In: 2013 21st Iranian Conference on Electrical Engineering (ICEE). pp. 1–5.

Sanchez, M. A., Castillo, O., Castro, J. R., 2015. Generalized Type-2 Fuzzy Systems for controlling a mobile robot and a performance comparison with Interval Type-2 and Type-1 Fuzzy Systems. Expert Systems with Applications 42 (14), 5904 – 5914.

Sarabakha, A., Kayacan, E., Dec 2016. Y6 Tricopter Autonomous Evacuation in an Indoor Environment Using Q-learning Algorithm. In: 2016 IEEE 55th Conference on Decision and Control (CDC). pp. 5992–5997.

Sun, J., Li, B., Jiang, Y., Wen, C.-y., 2016. A camera-based target detection and positioning uav system for search and rescue (sar) purposes. Sensors 16 (11).

Tai, K., El-Sayed, A.-R., Biglarbegian, M., Gonzalez, C. I., Castillo, O., Mahmud, S., 2016. Review of Recent Type-2 Fuzzy Controller Applications. Algorithms 9 (2).

Topalov, A. V., Kayacan, E., Oniz, Y., Kaynak, O., Aug 2009. Adaptive neuro-fuzzy control with sliding mode learning algorithm: Application to antilock braking system. In: 2009 7th Asian Control Conference. pp. 784–789.

Torres, M., Pelta, D. A., Verdegay, J. L., Torres, J. C., 2016. Coverage Path Planning with Unmanned Aerial Vehicles for 3D Terrain Reconstruction. Expert Systems with Applications 55, 441 – 451.

Valente, J., Sanz, D., Barrientos, A., Cerro, J. d., Ribeiro, Á., Rossi, C., 2011. An air-ground wireless sensor network for crop monitoring. Sensors 11 (6), 6088–6108.

Vanegas, F., Campbell, D., Eich, M., Gonzalez, F., 2016. Uav based target finding and tracking in gps-denied and cluttered environments. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2307–2313.

Wai, R. J., Muthusamy, R., 2013. Fuzzy-neural-network inherited sliding-mode control for robot manipulator including actuator dynamics. IEEE Transactions on Neural Networks and Learning Systems 24 (2), 274–287.

Wallace, L., Lucieer, A., Watson, C. S., 2014. Evaluating tree detection and segmentation routines on very high resolution uav lidar data. IEEE Transactions on Geoscience and Remote Sensing 52 (12), 7619–7628.

Wang, N., Er, M. J., Han, M., 2015. Dynamic tanker steering control using generalized ellipsoidal-basis-function-based fuzzy neural networks. IEEE Transactions on Fuzzy Systems 23 (5), 1414–1427.

Zhou, H., Kong, H., Wei, L., Creighton, D., Nahavandi, S., 2015. Efficient road detection and tracking for unmanned aerial vehicle. IEEE Transactions on Intelligent Transportation Systems 16 (1), 297–309.