Similarity-based Non-Singleton Fuzzy Logic Control for Improved Performance in UAVs

Changhong Fu[a,b], Andriy Sarabakha[a,b], Erdal Kayacan[a]
[a]School of Mechanical and Aerospace Engineering,
[b]ST Engineering-NTU Corp Laboratory,
Nanyang Technological University (NTU),
50 Nanyang Avenue, Singapore, 639798.
Email: changhongfu@ntu.edu.sg, andriy001@e.ntu.edu.sg, erdal@ntu.edu.sg

Christian Wagner[c,d], Robert John[d] and Jonathan M. Garibaldi[d]
[c]Institute of Computing and CyberSystems,
Michigan Technological University, Houghton, Michigan, USA.
[d]Lab for Uncertainty in Data and Decision Making (LUCID),
School of Computer Science,
University of Nottingham, Nottingham, United Kingdom.
Email: {christian.wagner, robert.john, jon.garibaldi}@nottingham.ac.uk

***Abstract*—As non-singleton fuzzy logic controllers (NSFLCs) are capable of capturing input uncertainties, they have been effectively used to control and navigate unmanned aerial vehicles (UAVs) recently. To further enhance the capability to handle the input uncertainty for the UAV applications, a novel NSFLC with the recently introduced similarity-based inference engine, i.e., Sim-NSFLC, is developed. In this paper, a comparative study in a 3D trajectory tracking application has been carried out using the aforementioned Sim-NSFLC and the NSFLCs with the standard as well as centroid composition-based inference engines, i.e., Sta-NSFLC and Cen-NSFLC. All the NSFLCs are developed within the robot operating system (ROS) using the C++ programming language. Extensive ROS Gazebo simulation-based experiments show that the Sim-NSFLCs can achieve better control performance for the UAVs in comparison with the Sta-NSFLCs and Cen-NSFLCs under different input noise levels.**

I. Introduction

Nowadays, unmanned aerial vehicles (UAVs) are widely used for various civilian applications [1], [2]. In most of these applications, classical control approaches, e.g. proportional-integral-derivative control [3] and sliding mode control [4], have been employed for UAVs to conduct autonomous flights. However, these well-known controllers require a precise dynamic model of the UAV and work under the assumption that significant internal as well as external uncertainties do not substantially affect the UAV systems. Achieving an accurate mathematical model for such complex aerial vehicles is often time-consuming [5]. In addition, the frequently-used sensors onboard the UAVs, e.g., camera, often lack precise modeling. Their measurements consist of numbers of uncertain, incomplete and possibly inaccurate information [6].

In the literature, fuzzy logic controllers (FLCs) are extensively used for the control and navigation of the UAVs. They are able to deliver adequate control and handle uncertainties without the requirement of an accurate mathematical UAV model. Among the different types of FLCs, singleton FLCs (SFLCs) are the most common FLCs used for the UAVs [7]. However, SFLCs are not capable of capturing input uncertainties effectively. Therefore, non-singleton FLCs (NSFLCs) are preferred as they can deal with the uncertainties by modeling the inputs as input fuzzy sets (FSs) [8]. In our previous paper [9], we investigated that applying two NSFLCs with standard and a novel centroid-based inference engines, i.e., Sta-NSFLC [8] and Cen-NSFLC [10], to control and stabilize the UAVs. The UAV flight results show that the Cen-NSFLCs can achieve better control performance than the Sta-NSFLCs. Furthermore, both the Sta-NSFLCs and Cen-NSFLCs outperform SFLCs under different levels of noise conditions. Despite that the NSFLCs are superior to the SFLCs, the NSFLCs applied for the UAV applications are still rare in comparison to the SFLCs.

In this work, a novel NSFLC with the similarity-based inference engine, i.e., Sim-NSFLC, is developed based on our recently introduced similarity-based non-singleton fuzzy logic system (NSFLS) inference engine [11] to navigate and guide a quadrotor UAV in a 3D trajectory tracking application. In this new approach, the firing strength of each rule is calculated by the similarity between the input and antecedent FSs instead of being calculated by the standard or centroid-based approach. The similarity-based inference engine is able to make the NSFLCs more sensitive to the changes of the input uncertainty. In [11], the Sim-NSFLS showed promising results in the well-known problem of Mackey-Glass time series predictions, i.e., the Sim-NSFLS outperformed the Sta-NSFLS and Cen-NSFLS under various noise conditions.

While this work is simulation-based, all the NSFLCs used in this work are developed within the robot operating system (ROS) [12] using C++ programming language to easily enable future, real-world experiments. The control performances of these NSFLCs are evaluated in the ROS Gazebo [13] environment, which provides a seamless connection for the developed algorithms between the simulation and real-world applications.

To the best of our knowledge, this is the first time in the literature that the similarity-based NSFLS inference is applied for control, i.e., as a Sim-NSFLC. The rest of this paper is structured as follows: Section II introduces quadrotor UAV dynamic model and control structure. Section III presents a brief background for the NSFLCs. Section IV evaluates control performances with all three NSFLCs under different input noise levels. Section V presents conclusions and future work.

II. Quadrotor UAV Dynamics and Control Structure

A. Quadrotor UAV Dynamics

In this work, the Parrot ARDrone 2 quadrotor UAV [14], as its Gazebo model shown in Fig. 1, is used to evaluate the NSFLCs. Let world inertial reference frame be $\mathcal{F}_I$, i.e., $\{\vec{x}_I, \vec{y}_I, \vec{z}_I\}$, and body frame be $\mathcal{F}_B$, i.e., $\{\vec{x}_B, \vec{y}_B, \vec{z}_B\}$. Figure 1 illustrates the UAV configuration and reference frames.

To achieve the translations and rotations of the UAV, the thrust of four rotors f_i, $i = 1, \ldots, 4$, are adjusted with various combinations. The thrust from each rotor is changed by controlling the angular speed w_i, $i = 1, \ldots, 4$ of the motor. The control input vector $\mathbf{c}$ of the UAV is represented as: $\mathbf{c} = \begin{bmatrix} T & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T$, where T is the total thrust along the $\vec{z}_B$ axis; τ_ϕ, τ_θ and τ_ψ are the moments acting on the $\vec{x}_B$, $\vec{y}_B$ and $\vec{z}_B$ axes. Then, the relationship between the control input $\mathbf{c}$ and angular speed ω_i, $i = 1, \ldots, 4$, is as follows [15]:

$$\begin{cases} T &= b\left(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2\right) \\ \tau_\phi &= \frac{\sqrt{2}}{2} bl\left(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2\right) \\ \tau_\theta &= \frac{\sqrt{2}}{2} bl\left(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2\right) \\ \tau_\psi &= d\left(-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2\right), \end{cases} \tag{1}$$

where b is the coefficient of propeller thrust, d is the coefficient of propeller drag and l is the UAV arm length.

Let the absolute position of the UAV be the three Cartesian coordinates of its mass center in the world frame $\mathcal{F}_I$, i.e, $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$, and its attitude be the three Euler angles, i.e, $\mathbf{o} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$, called roll, pitch and yaw, respectively. The time derivative of the absolute position is denoted as $\mathbf{v} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T = \begin{bmatrix} u & v & w \end{bmatrix}^T$, where $\mathbf{v}$ is the absolute velocity of the UAV's mass center in $\mathcal{F}_I$. Moreover, the time derivative of the attitude is $\boldsymbol{\omega} = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$, which is the angular velocity in $\mathcal{F}_I$, and the angular velocity in $\mathcal{F}_B$ is $\boldsymbol{\omega}_B = \begin{bmatrix} p & q & r \end{bmatrix}^T$. The dynamical model of the UAV is:

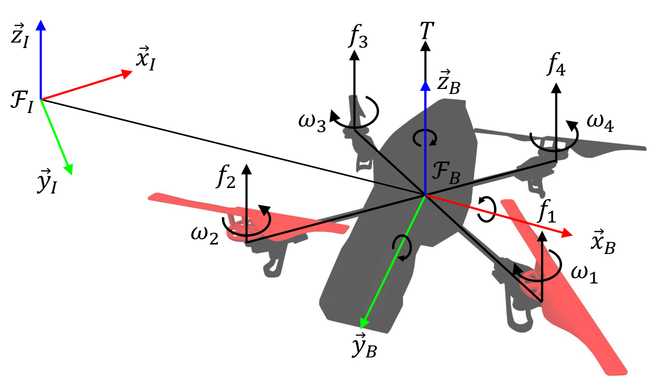


Fig. 1: Quadrotor UAV model with reference frames.

$$\begin{cases} \dot{x} = u & \dot{u} = \frac{1}{m}\left(c_\phi c_\psi s_\theta + s_\phi s_\psi\right) T \\ \dot{y} = v & \dot{v} = \frac{1}{m}\left(c_\phi s_\psi s_\theta - c_\psi s_\phi\right) T \\ \dot{z} = w & \dot{w} = \frac{1}{m} c_\phi c_\theta T - g \\ \dot{\phi} = p + s_\phi t_\theta q + c_\phi t_\theta r & \dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{1}{I_x}\tau_\phi \\ \dot{\theta} = c_\phi q - s_\phi r & \dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{1}{I_y}\tau_\theta \\ \dot{\psi} = \frac{s_\phi}{c_\theta} q + \frac{c_\phi}{c_\theta} r & \dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{1}{I_z}\tau_\psi, \end{cases} \tag{2}$$

where c_*, s_* and t_* denote $\cos *$, $\sin *$ and $\tan *$, m is the mass of the UAV, g is the gravity acceleration, i.e, $g = 9.81 m/s^2$, and $I = \mathrm{diag}(I_x, I_y, I_z)$ is the inertia matrix. As can be seen from the above dynamic equations, these equations are coupled, non-linear and the system to be controlled is under-actuated. Additionally, as discussed uncertainties in the real-world control of the UAV are inevitable. Hence, a fuzzy logic controller is utilized in this work instead of using a model-based linear controller.

B. Control Structure

The high-level NSFLC-based closed-loop control structure is illustrated in Fig. 2. It consists of two modules (shown with dashed rectangles): the position controller and the UAV. The position controller module includes three independent NS-FLCs, which take the desired position $\mathbf{p}_d = \begin{bmatrix} x_d & y_d & z_d \end{bmatrix}^T$ and current measured position $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ as the inputs, and then compute the control command $\mathbf{u}_d$, i.e., the desired roll ϕ_d, desired pitch θ_d angles and desired vertical velocity v_d^z. Specifically, considering the x-axis NSFLC as an example, the error e_x, i.e., $x_d - x$, the integral of the error $\int e_x$ and the derivative of the error de_x are calculated, and then the x-axis NSFLC outputs the desired roll ϕ_d. The detail on how

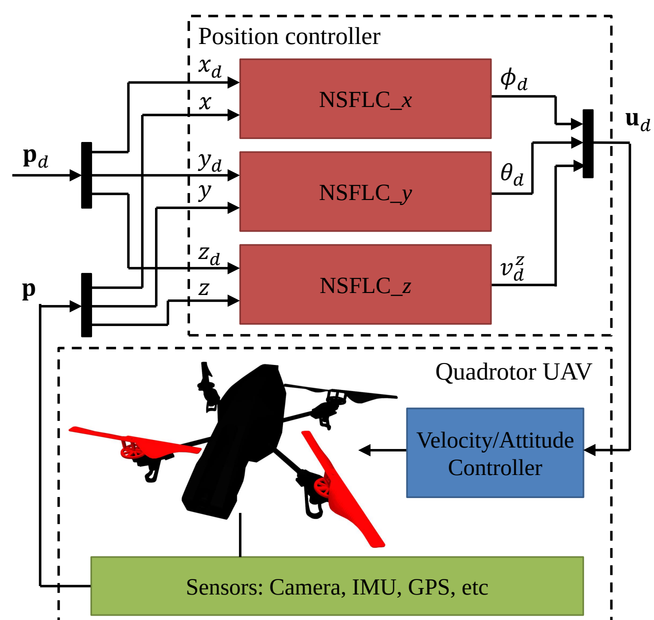


Fig. 2: The NSFLC-based closed-loop control structure for the navigation of the quadrotor UAV.

the NSFLC converts these errors to the desired command is introduced in Section III. The UAV module contains the low-level velocity/attitude controllers, the UAV system as well as the onboard sensors. The onboard sensors are used to measure the current UAV position and thus the input to the FLC.

III. Background of NSFLCs

A. Structure of NSFLC

A NSFLC includes fuzzifier, inference engine, rule base and defuzzifier [8]. Specifically, the NSFLC utilizes a non-singleton fuzzifier to model input uncertainties. In other words, the fuzzifier maps a crisp input to an input FS with a membership function (MF) around x' for handling the uncertainties from the actual input. In this work, a Gaussian distribution is employed for the fuzzifier:

$$\mu_X(x_i) = \exp\left[\frac{-(x_i - x_i')^2}{2\sigma_F^2}\right]. \tag{3}$$

where x_i' is the input crisp value and the mean value of the FS. σ_F is the spread of the FS. Larger values of the σ_F imply that more noise is expected in relation to the input data. It is noted that the crisp input can be a vector with multiple elements, as the three inputs e_x, $\int e_x$ and de_x. And each element in this vector is fuzzified with a Gaussian distribution.

In the literature, NSFLCs, which have been used for controlling UAVs, can be generally divided into two types based on different composition-based inference engines [9]: (I) the NSFLC with standard composition-based inference engine [8], i.e., Sta-NSFLC and (II) the NSFLC with centroid composition-based inference engine [10], i.e., Cen-NSFLC. In the Cen-NSFLC, the centroid of the FS intersection between the input and antecedent FSs is used for calculating the firing strength of each rule rather than the maximum of the intersection utilized in Sta-NSFLCs. The main motivation in this paper is to leverage an even more effective mechanism to integrate the input uncertainty into the inference engine, thereby making the NSFLCs more sensitive to the changes of the input uncertainty model in comparison with the Sta-NSFLC and Cen-NSFLC.

B. The standard NSFLC Inference Engine for UAV control

The general mapping between the inputs and outputs of the NSFLC is described in Fig. 3. To keep the description consistent with Section II-B, we still consider the x-axis NSFLC as an example. A triple-input, single-rule and single-output discrete NSFLC is considered, and the Mamdani implication is employed. Figure 3 illustrates the calculation from inputs (X_e, X_{de} and $X_{\int e}$), antecedents (A_e, A_{de} and $A_{\int e}$) and consequent (C) fuzzy sets to output (Y) of a Sta-NSFLC. Here, the crisp input is a vector which includes three elements, i.e., $\mathbf{x} = \begin{bmatrix} e & de & \int e \end{bmatrix}^T$, as discussed in the Section II-B. The e, de and $\int e$ are the members of the input FSs X_e, X_{de} and $X_{\int e}$, respectively. y is the member of the output FS (Y). Moreover, $\mu_{X_*}(*)$, $\mu_{A_*}(*)$, $\mu_C(y)$ and $\mu_Y(y)$

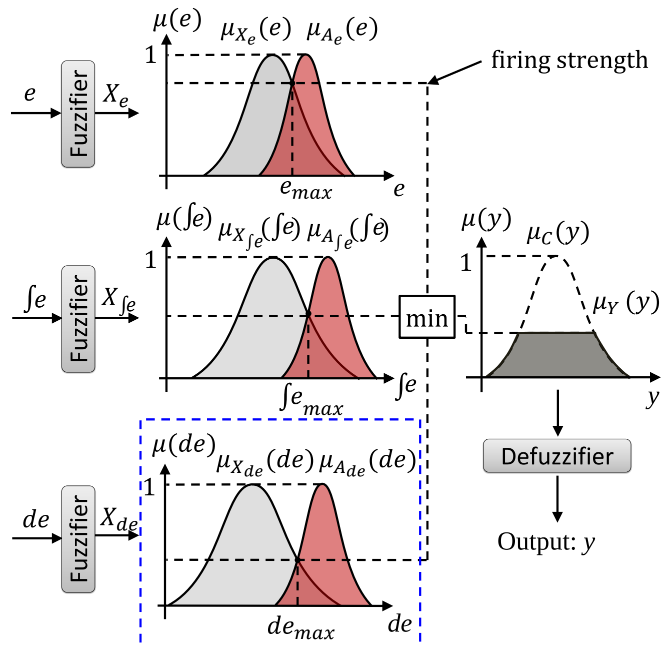


Fig. 3: Example of the calculation from inputs (X_*), antecedents (A_*) and consequent (C) fuzzy sets to output (Y) of a Sta-NSFLC.

be the membership functions (MFs) of X_*, A_*, C and Y, respectively. The defined rule is as follows:

IF e is A_e AND de is A_{de} AND $\int e$ is $A_{\int e}$ THEN y is C.

The input-output mapping of the Sta-NSFLC is:

$$\mu_Y(y) = \min[\mu_C(y),\ \min[\mu_e, \mu_{de}, \mu_{\int e}]], \tag{4}$$

where,

$$\begin{cases} \mu_e &= \max[\mu_{X_e}(e) \star \mu_{A_e}(e)] \\ \mu_{\int e} &= \max[\mu_{X_{\int e}}(\int e) \star \mu_{A_{\int e}}(\int e)] \\ \mu_{de} &= \max[\mu_{X_{de}}(de) \star \mu_{A_{de}}(de)], \end{cases}$$

$\mu_{X_*}(*) \star \mu_{A_*}(*)$ is the intersection of X_* and A_*.

The above equations show that the firing level of an antecedent is the maximum of its intersection with the input FS.

C. The NSFLC with Centroid-based Inference Engine (Cen-NSFLC)

To make the NSFLC more sensitive to the input uncertainty, the Cen-NSFLCs have been presented to control and stabilize the UAVs recently [9]. Taking the derivative of error de for example (as the blue rectangle shown in the Fig. 3), two different input FSs, i.e. X_{de}^1 and X_{de}^2, which are intersected with the same antecedent A_{de}, are considered, as shown in Fig. 4. Despite that the actual input FSs are different, the firing levels calculated by the standard approach in the Sta-NSFLC are the same in both cases, i.e., $\mu_{X_{de}}^1(de_{max}) = \mu_{X_{de}}^2(de_{max}) = \alpha$. While the centroid-based approach in the Cen-NSFLC has higher sensitivity to the shape of the intersection between the

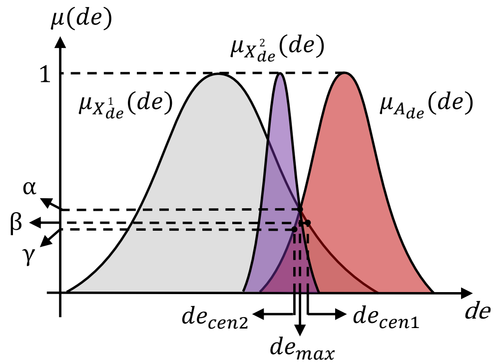


Fig. 4: The difference between the Cen-NSFLC and Sta-NSFLC. α, β and γ are the different firing levels.

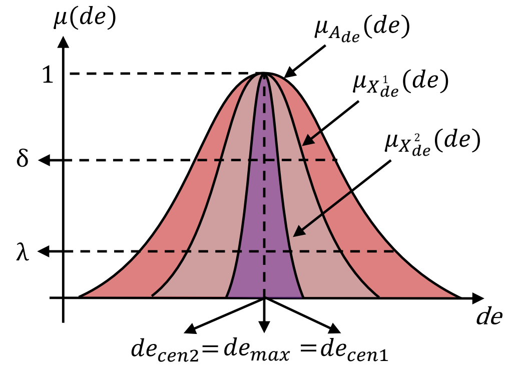


Fig. 5: The difference for the Sim-NSFLC, Cen-NSFLC and Sta-NSFLC. δ and λ are the different firing strengths of A_{de} for X_{de}^1 and X_{de}^2.

input FS and antecedent FS, i.e., these two various inputs with a different associated uncertainty distribution can generate two different firing levels, i.e., β and γ.

Considering a discrete FS X_{de} with a membership function $\mu_{X_{de}}(de_i)$, the centroid of X_{de} is defined as:

$$x_{cen}(X_{de}) = \frac{\sum_{i=1}^{n} de_i \mu_{X_{de}}(de_i)}{\sum_{i=1}^{n} \mu_{X_{de}}(de_i)}, \tag{5}$$

where n is the number of discretization levels (we set n=100 for our work) utilized in a discrete system.

The input-output mapping of the Cen-NSFLC is:

$$\mu_Y(y) = \min[\mu_C(y),\ \min[\mu_e, \mu_{de}, \mu_{\int e}]], \tag{6}$$

where,

$$\begin{cases} \mu_e & = \mu_{X_e \cap A_e}(x_{cen}(X_e \cap A_e)), \\ \mu_{\int e} & = \mu_{X_{\int e} \cap A_{\int e}}(x_{cen}(X_{\int e} \cap A_{\int e})), \\ \mu_{de} & = \mu_{X_{de} \cap A_{de}}(x_{cen}(X_{de} \cap A_{de})). \end{cases}$$

$x_{cen}(X_* \cap A_*)$ is the centroid of the intersection of an input X_* and an antecedent A_*. The aforementioned formulations show that the firing level of an antecedent is its membership degree at the centroid of the intersection with the input FS.

Although the Cen-NSFLC outperformed the Sta-NSFLC for the autonomous control and stabilization of the UAVs in our previous work [9]. Based on [11], we hypothesize that the performance of the NSFLCs can be further improved in our UAV tests by replacing the composition-based inference engine with the similarity-based inference engine.

D. The Similarity-based NSFLC (Sim-NSFLC)

We start by illustrating the rationale for the Sim-NSFLS as originally outlined in [11]. Figure 5 shows intersections of two different input FSs with one antecedent FS, although these two different input FSs have two different associated uncertainty distributions. The standard and centroid-based firing strengths of A_{de} for both X_{de}^1 and X_{de}^2 are the same, i.e, $\mu_{X_{de}}^1(de_{cen1}) = \mu_{X_{de}}^2(de_{cen2}) = \mu_{X_{de}}^1(de_{max}) = \mu_{X_{de}}^2(de_{max}) = 1$. Hence, a new NSFLS, which is more sensitive to the input uncertainty, is desirable. In our previous work [11], a novel NSFLS with the similarity-based inference engine, i.e., Sim-NSFLS, was used for the well-known problem of Mackey-Glass time series predictions. The prediction results showed that the Sim-NSFLS outperformed the Sta-NSFLS and Cen-NSFLS under different noise conditions.

As shown in the Fig. 5, δ and λ are two different firing levels for two different inputs based on the similarity-based approach. In this work, the Sim-NSFLC is developed to control and navigate the UAVs in a 3D trajectory tracking application.

Considering the input FS X_{de} and antecedent FS A_{de} with membership functions $\mu_{X_{de}}(de)$ and $\mu_{A_{de}}(de)$, the similarity between the X_{de} and A_{de} is defined based on the Jaccard similarity [16]:

$$s(X_{de}, A_{de}) = \frac{\int_{de \in X_{de}} \min(\mu_{X_{de}}(de), \mu_{A_{de}}(de))}{\int_{de \in X_{de}} \max(\mu_{X_{de}}(de), \mu_{A_{de}}(de))}, \tag{7}$$

In discrete domain, the above equation can be rewritten as:

$$s(X_{de}, A_{de}) = \frac{\sum_{i=1}^{n} \min(\mu_{X_{de}}(de), \mu_{A_{de}}(de))}{\sum_{i=1}^{n} \max(\mu_{X_{de}}(de), \mu_{A_{de}}(de))}, \tag{8}$$

where n is also the number of discretization levels (we set n=100 for our work).

The input-output mapping of the Sim-NSFLC is:

$$\mu_Y(y) = \min[\mu_C(y),\ \min[\mu_e, \mu_{de}, \mu_{\int e}]], \tag{9}$$

where,

$$\begin{cases} \mu_e & = s(X_e, A_e) \\ \mu_{\int e} & = s(X_{\int e}, A_{\int e}) \\ \mu_{de} & = s(X_{de}, A_{de}). \end{cases}$$

The above formulations show that the firing level is the similarity between the antecedent and the input FS. The following section investigates the control performance of the Sim-NSFLC and compare it with the composition-based NSFLCs, i.e., Sta-NSFLC and Cen-NSFLC.

IV. Simulation Studies

A. 3D trajectory generation

The 3D trajectory is defined according to the minimize snap property [17], which enables the real-time generation of an optimal trajectory through a sequence of 3D positions, thereby ensuring safe passage through specified environments as well as maintaining the constraints on accelerations and velocities. Similar to [18], some manoeuvrable flights were generated, e.g., descending and climbing straight lines as well as curves, the sharp turns between the straight lines and curves, to test the control performance of each NSFLC controller.

B. Fuzzifier, membership function and rule base

Different Gaussian MFs (with different standard deviations) are tested to evaluate the capture capability for the expected input uncertainty or noise in each of the NSFLC controllers. Each input variable, i.e., error, the integral of the error or the derivative of the error, has three MFs, and the output variable has five MFs. Table I shows the rule base of each NSFLC, where each abbreviation Z, N, P, S, B represents zero, negative, positive, small or big, respectively.

C. Intrinsic parameters of quadrotor UAV

These intrinsic parameters are determined based on the ones of a real Parrot AR Drone 2 quadrotor UAV, i.e., $b = 8.54 \times 10^{-6} N \cdot s^2$, $d = 1.6 \times 10^{-2} N \cdot m \cdot s^2$, $I_x = I_y = 0.007 kg \cdot m^2$, $I_z = 0.012 kg \cdot m^2$, $l = 0.18m$ and $m = 0.68kg$.

D. Noise generation and control performance evaluation

The Gaussian noise is defined by a noise generator [9], which injects the noise to the sensors of each UAV. The noise level is parameterized by its standard deviation σ_N. In addition, the control performance evaluation is conducted in terms of the mean squared error (MSE) of the 3D position.

TABLE I: Rule Base for all NSFLCs

Integral Error	Derivative Error	Proportional Error		
		N	Z	P
N	N	BP	BP	SP
	Z	BP	SP	Z
	P	SP	Z	SN
Z	N	BP	SP	Z
	Z	SP	Z	SN
	P	Z	SN	BN
P	N	SP	Z	SN
	Z	Z	SN	BN
	P	SN	BN	BN

E. Simulation results

In this work, eleven levels of noise and five instances of the NSFLCs with different input fuzzifications (i.e., different standard deviations for input MFs) are provided to evaluate the quadrotor UAV control performances using the aforementioned three NSFLCs. Each combination of the noise and fuzzifer for one NSFLC is evaluated for 30 times. Figure 6 shows the example of three UAV flights with the same level of fuzzifier (σ_F=1.0) under three different levels of noise (σ_N=0.0, 0.5 and 1.0). Table II shows the average MSE of the 3D position. As shown in Table II, the Cen-NSFLCs outperform the Sta-NSFLCs, and the control performances of the Sim-NSFLCs are better than both the Cen-NSFLCs and Sta-NSFLCs. In addition, the larger values of the σ_F for the fuzzifier can assist the NSFLCS to achieve better performances. Figure 7 also clearly shows the control performance differences among the Sta-NSFLC, Cen-NSFLC and Sim-NSFLC.

A demonstration video related to our work can be found at: https://youtu.be/NVfgz38RFuA.

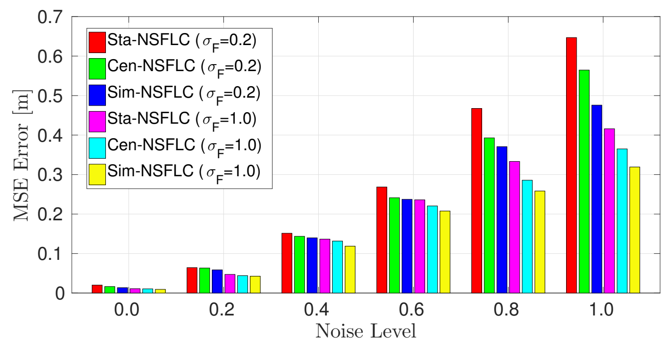
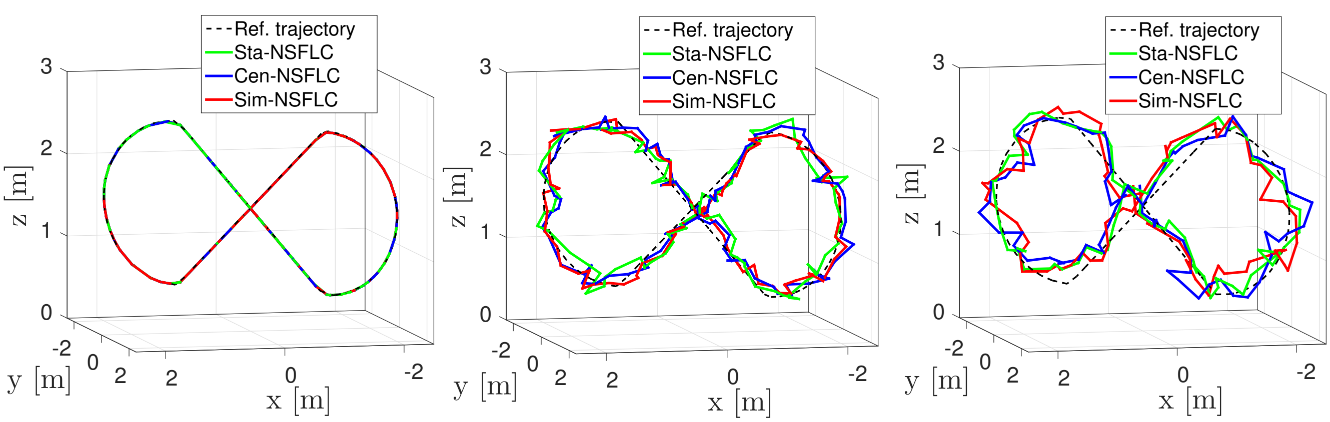


Fig. 7: Control performances of the Sta-NSFLC, Cen-NSFLC and Sim-NSFLC in 3D trajectory tracking task.



(a) σ_F=1.0, σ_N=0.0 (b) σ_F=1.0, σ_N=0.5 (c) σ_F=1.0, σ_N=1.0

Fig. 6: Example of three UAV flights with the same input fuzzification (σ_F=1.0) under three different levels of noise.

TABLE II: Average MSE of 3D Position (Unit: Meter)

NSFLC / Noise Level (σ_N)	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Sta-NSFLC (σ_F=0.2)	0.0202	0.0267	0.0643	0.1065	0.1514	0.2079	0.2686	0.4069	0.4674	0.4938	0.6469
Cen-NSFLC (σ_F=0.2)	0.0165	0.0251	0.0634	0.0969	0.1436	0.1910	0.2413	0.3781	0.3929	0.4657	0.5646
Sim-NSFLC (σ_F=0.2)	**0.0136**	**0.0232**	**0.0587**	**0.0866**	**0.1398**	**0.1862**	**0.2373**	**0.3156**	**0.3707**	**0.4348**	**0.4758**
Sta-NSFLC (σ_F=0.4)	0.0168	0.0258	0.0642	0.0921	0.1423	0.2026	0.2619	0.3738	0.4650	0.4917	0.6271
Cen-NSFLC (σ_F=0.4)	0.0132	0.0240	0.0603	0.0916	0.1381	0.1903	0.2408	0.3482	0.3822	0.4476	0.5190
Sim-NSFLC (σ_F=0.4)	**0.0117**	**0.0229**	**0.0528**	**0.0860**	**0.1233**	**0.1729**	**0.2315**	**0.3132**	**0.3602**	**0.4006**	**0.4603**
Sta-NSFLC (σ_F=0.6)	0.0121	0.0243	0.0626	0.0916	0.1412	0.1978	0.2617	0.3637	0.3843	0.4586	0.5589
Cen-NSFLC (σ_F=0.6)	0.0119	0.0220	0.0592	0.0901	0.1362	0.1874	0.2329	0.3309	0.3783	0.4472	0.5059
Sim-NSFLC (σ_F=0.6)	**0.0112**	**0.0204**	**0.0502**	**0.0857**	**0.1225**	**0.1646**	**0.2241**	**0.2448**	**0.3572**	**0.3848**	**0.4478**
Sta-NSFLC (σ_F=0.8)	0.0114	0.0237	0.0541	0.0914	0.1411	0.1615	0.2372	0.3234	0.3645	0.4581	0.4986
Cen-NSFLC (σ_F=0.8)	0.0118	0.0213	0.0509	0.0884	0.1348	0.1450	0.2326	0.2941	0.3302	0.3960	0.4694
Sim-NSFLC (σ_F=0.8)	**0.0103**	**0.0200**	**0.0434**	**0.0785**	**0.1205**	**0.1403**	**0.2117**	**0.2372**	**0.3225**	**0.3521**	**0.4017**
Sta-NSFLC (σ_F=1.0)	0.0109	0.0232	0.0472	0.0844	0.1366	0.1610	0.2361	0.3143	0.3332	0.3977	0.4160
Cen-NSFLC (σ_F=1.0)	0.0106	0.0211	0.0440	0.0805	0.1317	0.1405	0.2206	0.2610	0.2857	0.3543	0.3650
Sim-NSFLC (σ_F=1.0)	**0.0093**	**0.0192**	**0.0426**	**0.0769**	**0.1187**	**0.1259**	**0.2077**	**0.2247**	**0.2584**	**0.3082**	**0.3192**

V. Conclusion and Future Work

In this work, a novel NSFLC with similarity-based inference engine has been developed and deployed to control simulated UAVs in the 3D trajectory tracking application. A comprehensive comparison and evaluation has been carried out with three different types of NSFLCs, i.e., Sta-NSFLC, Cen-NSFLC and the novel NSFLC (Sim-NSFLC), under different levels of input uncertainty, i.e., noise. The aim of this work was not only to evaluate the control performances among these three NSFLCs, but also to explore better NSFLCs for the real-world UAV applications. All the NSFLCs are programmed in the C++ language and evaluated in the ROS and Gazebo environment. The extensive simulation tests show that the Sim-NSFLC can obtain better control performances compared to the Sta-NSFLC and Cen-NSFLC, especially at the higher input noise levels. Moreover, the different input fuzzifications can achieve the various capabilities for capturing input uncertainties. In other words, the higher input fuzzification has more capability to handle higher level input noise. These results support the results in [11].

For future work, the experiments on real-world quadrotor UAVs will be conducted and we will extend this approach to different Type-2 FLCs to control the quadrotor UAVs, and compare their performances under different input noise levels. Finally, the novel NSFLC architecture provides improved capacity to explore detailed input uncertainty models (captured in the input fuzzy sets), thus a key aspect of our future work is to develop improved techniques to appropriately capture real-world input noise/uncertainty in the input MFs of the FLCs.

Acknowledgment

This research work was partially supported by the ST Engineering-NTU Corporate Lab through the NRF corporate lab@university scheme. This work was also partially funded by the RCUK's EP/M02315X/1 From Human Data to Personal Experience grant.

References

[1] Y. Lin and S. Saripalli, "Sense and avoid for Unmanned Aerial Vehicles using ADS-B," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6402–6407.

[2] M. Mueller, G. Sharma, N. Smith, and B. Ghanem, "Persistent Aerial Tracking system for UAVs," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1562–1569.

[3] J. Pestana, I. Mellado-Bataller, C. Fu, J. L. Sanchez-Lopez, I. F. Mondragon, and P. Campoy, "A general purpose configurable navigation controller for micro aerial multirotor vehicles," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2013, pp. 557–564.

[4] J. R. Hervas, E. Kayacan, M. Reyhanoglu, and H. Tang, "Sliding mode control of fixed-wing uavs in windy environments," in *13th International Conference on Control Automation Robotics Vision (ICARCV)*, 2014, pp. 986–991.

[5] T. Lee, M. Leok, and N. H. McClamroch, "Nonlinear Robust Tracking Control of a Quadrotor UAV on SE(3)," *Asian Journal of Control*, vol. 15, no. 2, pp. 391–408, 2013.

[6] C. Fu, A. Carrio, and P. Campoy, "Efficient visual odometry and mapping for Unmanned Aerial Vehicle using ARM-based stereo vision pre-processing system," in *Unmanned Aircraft Systems (ICUAS), International Conference on*, 2015, pp. 957–962.

[7] C. Fu, M. A. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy, "Monocular Visual-Inertial SLAM-Based Collision Avoidance Strategy for Fail-Safe UAV Using Fuzzy Logic Controllers," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 513–533, 2014.

[8] J. Mendel, *Uncertain rule-based fuzzy logic system: introduction and new directions*. Upper Saddle River, NJ, USA, Prentice-Hall, 2001.

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

[10] A. Pourabdollah, C. Wagner, J. H. Aladi, and J. M. Garibaldi, "Improved uncertainty capture for nonsingleton fuzzy systems," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 6, pp. 1513–1524, 2016.

[11] C. Wagner, A. Pourabdollah, J. McCulloch, R. John, and J. M. Garibaldi, "A similarity-based inference engine for non-singleton fuzzy logic systems," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2016, pp. 316–323.

[12] "Robot Operating System (ROS)," http://www.ros.org/, 2016.

[13] "Gazebo," http://www.gazebosim.org/, 2016.

[14] "ARDrone Parrot 2," https://www.parrot.com/, 2016.

[15] R. Mahony, V. Kumar, and P. Corke, "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[16] P. Jaccard, "Distribution de la flore alpine dans le bassin des Dranses et dans quelques regions voisines," *Bulletin de la Societe Vaudoise des Sciences Naturelles*, vol. 37, pp. 241–272, 1901.

[17] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[18] E. Kayacan and R. Maslim, "Type-2 Fuzzy Logic Trajectory Tracking Control of Quadrotor VTOL Aircraft With Elliptic Membership Functions," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 1, pp. 339–348, 2017.