Journal of Intelligent and Fuzzy Systems 1 (2021) 1–16
IOS Press


# FLC-ROS: A Generic and Configurable ROS Package for Developing Fuzzy Logic Controllers

Ali Karimoddini [a,*], Abel Hailemichael [a], and Mo Jamshidi [b]

[a] *Department of Electrical and Computer Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC 27411 USA* **

[b] *Department of Electrical and Computer Engineering, The University of Texas, San Antonio, TX 78249 USA*

**Abstract.** Fuzzy logic controllers can handle complex systems by incorporating expert's knowledge in the absence of formal mathematical models. Further, fuzzy logic controllers can effectively capture and accommodate uncertainties that are inherent in real-world controlled systems. On the other hand, *Robot Operating System* (ROS) has been widely used for many robotic applications due to its modular structure and efficient message-passing mechanisms for the integration of system's components. For this reason, Robot Operating System is an ideal tool for developing software stacks for robotic applications. This paper develops a generic and configurable Robot Operating System package for the implementation of fuzzy logic controllers, particularly type-1 and interval type-2, which are based on either Mamdani or Takagi-Sugeno-Kang fuzzy inference mechanisms. This is achieved by employing a systematic object-oriented approach using the *Unified Model Language* (UML) to implement the fuzzy inference system as a single class that is composed of fuzzifier, inference, and defuzzifier classes. The deployment of the developed Robot Operating System package is demonstrated by implementing an interval type-2 fuzzy logic control of an *Unmanned Aerial Vehicle* (UAV).

Keywords: Robot Operating System, Unified Model Language, Type-1 Fuzzy Logic Control, Interval Type-2 Fuzzy Logic System, TSK, Mamdani, ROS, UAV, Unmanned Aerial Vehicles.

## 1. Introduction

A major challenge in any control system is to deal with uncertainties that exist in real-world systems due to sensor-reading inaccuracy, noises in the environment, inaccuracy of model parameters, and external disturbances. To handle these situations, classical robust control systems often require an accurate model of the system and bounds of disturbances which may not be always available. In contrast, *artificial intelligent* (AI) techniques, including but limited to *Artificial Neural Network* (ANN), *Genetic Algorithm* (GA), *Support Vector Machine* (SVM), *Expert Systems* (ES), *Neural Language Programming* (NLP), *Reinforcement Learning* (RL), *Fuzzy Logic (FL)*, as well as, hybrid approaches (combination of AI techniques), are capable of handling control problems when the system's model is unknown or far too complex to accurately model [1–4]. In this vein, *Fuzzy Logic Controllers*

*Corresponding author: A. Karimoddini, Tel: +13362853313, akarimod@ncat.edu.

**This work is supported by the NSF under the award number 1832110 and Air Force Research Laboratory and Office of the Secretary of Defense under agreement number FA8750-15-2-0116.

(FLCs) use approximate reasoning to deal with complex ill-defined systems with uncertain models and stochastic behaviors and disturbances [5, 6]. FLCs use fuzzy logic for computing the control signals in order to control a system. Fuzzy logic is an approach for knowledge representation and symbolic inference, computation, and reasoning based on human-like linguistic expressions with degrees of truth [7]. By using membership functions and rule-based inference mechanisms, FLCs are exceptionally powerful in mimicking human decision-making in the absence of formal mathematical models for a system. Furthermore, FLCs (particularly type-2 FLCs [8]) are capable of effectively capturing and handling uncertainties [9, 10]. The ability of FLCs to handle dynamic behaviors of complex systems, without knowing much about their mathematical model, has made them suitable for the design of real-world robotic systems [11] and industrial control systems such as such as the control of liquid-level process [12], control of unmanned aerial vehicles [13], traffic signal control [14], fault detection [15], autonomous vehicle applications [16], maritime radar detection mechanism [17], human behavior modeling [18], testing and evaluation of autonomous vehicles [19, 20], and pattern recognition [21]. However, the structure of a fuzzy logic controller is relatively more complex than many other commonly-used controllers, e.g, PID controllers, and often involves several procedures including fuzzification, inference, type-reduction, and defuzzifcation, which requires a lot of effort for the development, implementation, testing, and verification of the FLCs [13, 22, 23]. This has hindered control application developers from utilizing the advantages of FLCs, particularly type-2 FLCs.

In this paper, we develop a user-friendly development tool in order to ease the implementation of FLCs and facilitate fast prototyping FLC-based robotic and control applications. The developed tool will be provided as a *Robot Operating System* (ROS) package. Our choice of ROS is motivated by the fact that its modular structure and enhanced robust communication and system integration mechanisms enable the rapid and reliable development of robotic and autonomous systems [24–26]. As such, ROS has been widely adopted by the robotic community as the premier development platform for developing software stacks for different robotic applications [27–29]. While some work has been done to develop ROS packages for the implementation of type-1 Mamdani FLCs [30, 31], they are limited to a particular class of FLCs and do not support the development of different inference systems such as *Takagi-Sugeno-Kang* (TSK) fuzzy inference mechanisms, and are developed for a specific application. Therefore, their deployment and generalization for other use cases and applications are not straightforward. Hence, the challenge to find a suitable ROS package, which provides the required functionality and flexibility for the design and implementation of FLCs is still open. To overcome this challenge, this paper provides, to the best of our knowledge, the first generic and configurable ROS package that eases the real-time implementation of type-1 and interval type-2 FLCs, which are based on both Mamdani and TSK inference mechanisms.

The contributions of this paper are as follows:

- We have developed systematic configuration techniques, software architecture, and algorithms that enable the implementations of a generic and configurable FLC ROS packages.
- To implement multiple FLCs in a single application, this paper presents a systematic object-oriented development approach representing the entire fuzzy logic system by a single class, which is composed of fuzzifier and inference as well as defuzzifier classes. Algorithms and structures that enable configuration and deployment of *multi-input-multi-output* (MIMO) Mamdani or TSK fuzzification, inference mechanisms, and defuzzification operations are developed and implemented.
- To enable the development of computationally effective real-time interval type-2 FLCs, the developed FLC ROS package employs the uncertainty bounds [32, 33] and Nie-Tan [34] output processing techniques.
- The developed package constitutes a user-friendly *graphical user interface* (GUI) and configuration file, allowing configuration of MIMO fuzzy

logic control parameters such as number of inputs, number of outputs, rules, membership functions, inference methods, type reduction, and defuzzification. The package assists the implementation of FLCs within the ROS environment without demanding the detailed mathematical knowledge of type-1 and type-2 FLCs.

– The effectiveness of the developed FLC ROS package is demonstrated via a *software-in-the-loop* (SITL) for implementing an interval type-2 TSK fuzzy logic controller for the altitude control of a quadcopter UAV in ROS Gazebo simulation environment.

The rest of this paper is organized as follows. Section 2 briefly provides the necessary preliminaries and backgrounds on fuzzy control systems, ROS, and UML. Section 3 discusses the developed FLC ROS package's structure and functionalities. Section 4 describes the deployment of the developed ROS package within a simulation environment for the altitude control of a quadcopter UAV. The paper is concluded in Section 5.

## 2. PRELIMINARIES

This section presents a brief review of fuzzy set theory and fuzzy logic systems. A fuzzy set is a set whose elements have degrees of membership. Fuzzy logic controllers employ rule-based inference techniques that use fuzzy sets for making control decisions. The development process of FLCs includes the processes of fuzzification, inference, and defuzzification. By using *membership functions* (MFs), the fuzzification process assigns membership values to crisp inputs. The inference process maps fuzzy inputs to fuzzy outputs according to pre-defined rules. Finally, it is the defuzzification process that generates crisp outputs from aggregated fuzzy outputs.

Based on the employed type of fuzzy sets, FLCs may be classified as type-1, type-2, or type-*n*. Also, FLCs may be classified based on the type of the employed inference mechanism such as Mamdani [35] or TSK [36].

### 2.1. Type-1 Fuzzy Logic Systems

**Definition 1.** *Type-1 Fuzzy Set*

*A type-1 fuzzy set is composed of elements of the set's domain, $x \in X$, and their corresponding membership values $(x, \mu_A(x))$ for which $\mu_A(x) \in [0, 1]$. Formally, a type-1 fuzzy set can be defined as:*

$$A = \{(x, \mu_A(x)) \mid \forall x \in X, \mu_A(x) \in [0,1]\} = \sum_{x \in X} (x, \mu_A(x)) \quad (1)$$

*where $\sum$ is the collection of elements of the set.*

An example of a type-1 fuzzy set with its corresponding membership function is shown in Fig. 1.

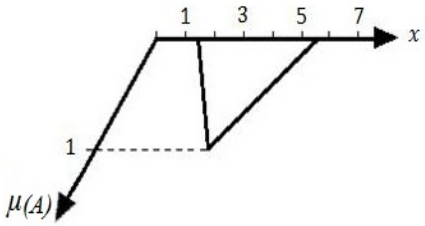



Fig. 1. A type-1 fuzzy set.

In type-1 Mamdani fuzzy inference systems [35], crisp inputs are initially fuzzified using predefined type-1 membership functions. The fuzzified inputs are then mapped to type-1 output fuzzy sets using the mamdani fuzzy inference process. The output fuzzy sets are finally aggregated and defuzzified to generate crisp outputs. A typical MIMO type-1 fuzzy logic controller is shown in Fig. 2.

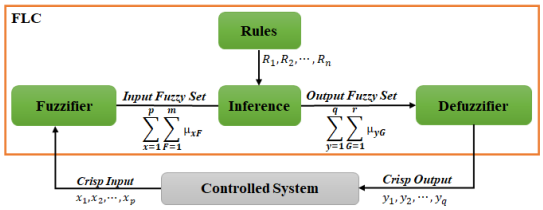



Fig. 2. A type-1 FLC structure.

Type-1 FLCs employ type-1 fuzzy sets on the antecedent and consequent sides of their rules. An exam-

ple a rule with $p$ inputs and $q$ outputs in a type-1 FLC with Mamdani fuzzy inference system can be stated as:

$$R^{\ell}\text{: } \textbf{\textit{IF}}\ x_1 \text{ is } {F_1}^{\ell}, \text{ and } x_2 \text{ is } {F_2}^{\ell}, \cdots, \text{and } x_p \text{ is } {F_p}^{\ell}$$
$$\textbf{\textit{THEN}}\ y_1 \text{ is } {G_1}^{\ell}, y_2 \text{ is } {G_2}^{\ell}, \cdots, y_q \text{ is } {G_q}^{\ell} \quad (2)$$

where $R^{\ell}$ is the $\ell$th rule, ${F_p}^{\ell}$ is the activated antecedent type-1 fuzzy set for input channel $x_p$, and ${G_q}^{\ell}$ is the activated consequent type-1 fuzzy set for output channel $y_q$, $m$ is the number of membership functions for a given input, and $r$ is the number of membership functions for a given output. The firing level for the $\ell$th rule is the t-norm of these fuzzy values obtained from the antecedent membership functions as indicated by the *Rule Set*. For each of the rules, the output fuzzy set can be found as the t-norm of the firing level and the output fuzzy set. The aggregated output fuzzy set can be found as the s-norm of the output fuzzy sets. Finally, using the centroid defuzzification as one of the most common defuzzification methods, we can calculate the crisp value of each output of a Mamdani fuzzy inference system as:

$$y_k = \frac{\sum_{i=1}^{n} \mu_k(x_{k,i}) y_{k,i}}{\sum_{i=1}^{n} \mu_k(x_{k,i})} \quad (3)$$

where $\mu_k$ is the membership function of the aggregated output $y_k$, $n$ is the number of samples in the aggregated output fuzzy set of the $k$th output channel, and $y_{k,i}$ is the $i$th sample.

In type-1 TSK fuzzy control systems [37], rule outputs are a function of the rule inputs. As an example, the rules for a TSK FLC with a $p$ inputs and $q$ outputs can be stated as:

$R^{\ell}$: ***IF*** $x_1$ *is* ${F_1}^{\ell}$, *and* $x_2$ *is* ${F_2}^{\ell}, \cdots, and x_p$ *is* ${F_p}^{\ell}$
**THEN**

$$y_1^{\ell} = c_{0,1}^{\ell} + c_{1,1}^{\ell} x_1 + \cdots + c_{p,1}^{\ell} x_p,$$
$$y_2^{\ell} = c_{0,2}^{\ell} + c_{1,2}^{\ell} x_1 + \cdots + c_{p,2}^{\ell} x_p,$$
$$\cdots$$
$$y_q^{\ell} = c_{0,q}^{\ell} + c_{1,q}^{\ell} x_1 + \cdots + c_{p,q}^{\ell} x_p,$$

where $R^{\ell}$ is the $\ell$th rule; ${F_p}^{\ell}$ is the activated antecedent type-1 fuzzy set for input channel $x_p$; and $c_{0,q}^{\ell}$, $c_{1,q}^{\ell}, \cdots, c_{p,q}^{\ell}$ are crisp TSK output coefficients for outputs $y_q^{\ell}$. The outputs of type-1 TSK fuzzy rules are crisp. To obtain the final (aggregated) crisp value of $k$th output, a defuzzification process could be performed using the weighted average technique as:

$$y_k = \frac{\sum_{i=1}^{\ell} f^{\ell} y_k^{\ell}}{\sum_{i=1}^{\ell} f^{\ell}} \quad (4)$$

where $f^{\ell}$ is the firing level for rule $\ell$ and $y_k^{\ell}$ is the consequent for output $k$ of the $\ell$th TSK rule.

### 2.2. Type-2 Fuzzy Logic Systems

Even though type-1 fuzzy sets enable linguistic computing by assigning a degree of membership to all their elements, they are not capable of quantifying the level of uncertainty in the degree of membership. It is, however, possible to successfully capture and quantify uncertainties by extending type-1 fuzzy sets to type-2 fuzzy sets [38, 39]. Type-2 fuzzy sets can be defined as follows:

**Definition 2.** *Type-2 Fuzzy Set*

*A type-2 fuzzy set is composed of triples* $((x,u), \mu_{\tilde{A}}(x,u))$ *in which secondary membership,* $\mu_{\tilde{A}}(x,u)$ *is defined for each member of domain* $x \in X$ *with the primary membership value,* $u \in J_x$ *(*$J_x$ *is the range of primary membership for a given* $x$*). Mathematically, a type-2 fuzzy set is defined as follows:*

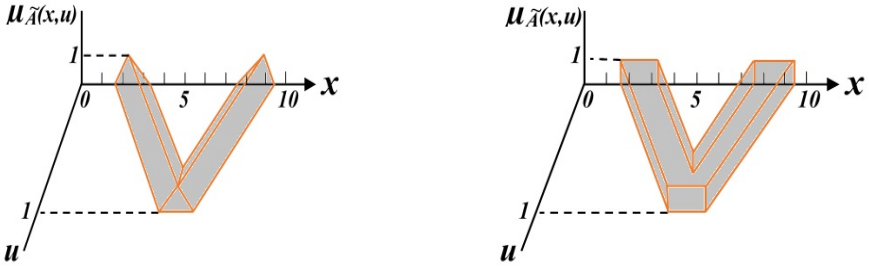
$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) | \ \forall x \in X, \forall u \in J_x \subseteq [0,1],$$
$$\mu_{\tilde{A}}(x,u) \in [0,1]\} = \sum_{u \in J_x} \sum_{x \in X} ((x,u), \mu_{\tilde{A}}(x,u)) \quad (5)$$

Fig. 3(a) shows a simple type-2 fuzzy set for a case that $X$ and $J_x$ are connected sets, and $\mu_{\tilde{A}}$ is a continuous function.

Type-2 FLCs capture input and output uncertainties by employing type-2 fuzzy sets on their antecedent and consequent parts of rules [40, 41]. Use of interval type-2 or $\alpha$-plane based FLCs can significantly reduce the computational complexity, facilitating real-time implementations [9, 42, 43].

**Definition 3.** *Interval Type-2 Fuzzy Set (IT2 FS)*

*An interval type-2 fuzzy set is a type-2 fuzzy set with*




(a) (b)

Fig. 3. (a) A type-2 fuzzy set, (b) An interval type-2 fuzzy set.

*secondary grade values set to unity. It is defined as:*

$$\tilde{A} = \{((x,u),1) | \forall x \in X, \forall u \in J_x \subseteq [0,1]\} = \sum_{u \in J_x} \sum_{x \in X} ((x,u),1) \quad (6)$$

Fig. 3(b) shows an example of an interval type-2 fuzzy set. Compared to type-1 fuzzy sets, interval type-2 fuzzy sets have more degrees of freedom in the form of upper and lower membership functions to better capture uncertainties, while generating smoother control surfaces [10, 44]. On the other hand, compared to type-2 FLSs, the use of interval type-2 fuzzy sets significantly reduces the computation costs, while maintaining major advantages of type-2 FLSs [9, 9, 45].

Both the rule antecedent and consequent sets of a type-2 Mamdani FLCs are type-2 fuzzy sets. An example of a $p$-input and $q$-output type-2 FLC rule can be stated as:

$$R^{\ell}: \textbf{\textit{IF}}\ x_1 \textit{ is } \tilde{F_1}^{\ell}, \textit{ and } x_2 \textit{ is } \tilde{F_2}^{\ell}, \cdots, \textit{and } x_p \textit{ is } \tilde{F_p}^{\ell}$$
$$\textbf{\textit{THEN}}\ y_1 \text{ is } \tilde{G_1}^{\ell}, y_2 \text{ is } \tilde{G_2}^{\ell}, \cdots, y_q \text{ is } \tilde{G_q}^{\ell} \quad (7)$$

where $R^{\ell}$ is the $\ell$th rule, $\tilde{F_p}^{\ell}$ is the activated antecedent type-2 fuzzy set for input channel $x_p$, and $\tilde{G_q}^{\ell}$ is the activated consequent type-2 fuzzy set for output channel $y_q$.

For type-2 Mamdani FLCs, a computationally expensive type reduction of the aggregated output fuzzy set is required prior to the defuzzification process. With an effort of easing this computation burden, several techniques have been previously developed, one of which is the Nie-Tan type reduction mechanism, which is applicable to interval type-2 FLCs [34]. With reasonable accuracy, the Nie-Tan type reduction mechanism is the least computationally expensive [45]. As stated in Equation 8, to compute the type reduced set, this method uses the average of the upper and lower membership functions of an aggregate interval type-2 output fuzzy set, $\tilde{c}_k$, for the $k$th output.

$$\mu_k^*(\tilde{y}_k) = \frac{1}{2}(\overline{\mu_{\tilde{c}_k}}(\tilde{y}_k) + \underline{\mu_{\tilde{c}_k}}(\tilde{y}_k)) \quad (8)$$

where $\mu^*$ is the membership function of the type reduced set, $\overline{\mu_{\tilde{c}}}(\tilde{y}_k)$ and $\underline{\mu_{\tilde{c}}}(\tilde{y}_k)$ are respectively the upper and lower membership functions of the aggregated output fuzzy set $\tilde{c}_k$ for the output $y_k$. Then, the centroid of the generated IT2 fuzzy set can be computed using Equation 9.

$$y_k = \frac{\sum_{i=1}^{n} \mu_k^*(\tilde{y}_{k,i}) \tilde{y}_{k,i}}{\sum_{i=1}^{k} \mu_k^*(\tilde{y}_{k,i})} \quad (9)$$

Instead of IT2 Mamdani FLCs, we can use IT2 TSK FLCs. Similar to type-1 TSK FLC rules, the rule outputs of IT2 TSK FLCs are functions the inputs. In this case, to capture inputs' uncertainty, we use Type-2 fuzzy sets for the antecedent MFs. On the other hand, for capturing output uncertainty, the consequent coefficients can be type-1 fuzzy sets [33]. An example of an IT2 TSK FLC rule with $p$ inputs and $q$ outputs can be stated as:

$$R^{\ell}: \textbf{\textit{IF}}\ x_1 \textit{ is } \tilde{F_1}^{\ell}, \textit{ and } x_2 \textit{ is } \tilde{F_2}^{\ell}, \cdots, \textit{and} x_p \textit{ is } \tilde{F_p}^{\ell}$$
$$\textbf{\textit{THEN}}$$
$$y_1^{\ell} = c_{0,1}^{\ell} + c_{1,1}^{\ell} x_1 + \cdots + c_{p,1}^{\ell} x_p,$$
$$y_2^{\ell} = c_{0,2}^{\ell} + c_{1,2}^{\ell} x_1 + \cdots + c_{p,2}^{\ell} x_p,$$
$$\cdots$$
$$y_q^{\ell} = c_{0,q}^{\ell} + c_{1,q}^{\ell} x_1 + \cdots + c_{p,q}^{\ell} x_p,$$

where $R^{\ell}$ is the $\ell$th rule; $\tilde{F}_p^{\ell}$ is the activated antecedent interval type-2 fuzzy set for input channel $x_p$; $c^{\ell}_{0,q}, c^{\ell}_{1,q}, \cdots, c^{\ell}_{p,q}$ are type-1 TSK rule output coefficients for outputs $y_q$. For reducing the computation burden, a computationally effective interval type-2 TSK output processing technique may be used [32]. This output processing technique approximates the upper and lower bounds of the weighted outputs, $y_l$ and $y_r$, by using only the upper and lower bounds of the rule firing levels and output coefficients. The upper and lower bounds of the firing levels can be computed using Equations 10 and 11, respectively.

$$\underline{f^{\ell}} = \underline{\tilde{F}_1^{\ell}}(x_1) * \underline{\tilde{F}_2^{\ell}}(x_2) * \cdots * \underline{\tilde{F}_p^{\ell}}(x_p) \quad (10)$$

$$\overline{f^{\ell}} = \overline{\tilde{F}_1^{\ell}}(x_1) * \overline{\tilde{F}_2^{\ell}}(x_2) * \cdots * \overline{\tilde{F}_p^{\ell}}(x_p) \quad (11)$$

The upper and lower bounds of the output, $y_l$ and $y_r$, can be estimated by calculating and averaging their upper and lower bounds values, $\underline{y_l}$, $\overline{y_l}$, $\underline{y_r}$, and $\overline{y_r}$ as shown in Fig. 4. The inner upper and lower bounds ($\overline{y_l}$ and $\underline{y_r}$) of $y_l$ and $y_r$ are calculated as follows:

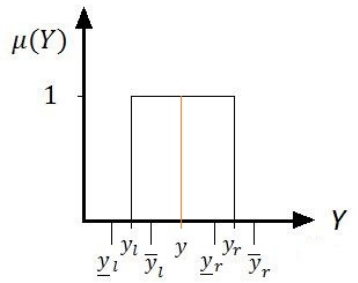



Fig. 4. Visual representation of output uncertainty bounds.

$$\overline{y_l} = \min\{y_{ll}, y_{ul}\} \quad (12)$$

$$\underline{y_r} = \max\{y_{lr}, y_{ur}\} \quad (13)$$

where,

$$y_{ll} = \frac{\underline{f^i} y_l^1 + \cdots + \underline{f^M} y_l^M}{\underline{f^i} + \cdots + \underline{f^M}} \quad (14)$$

$$y_{ul} = \frac{\overline{f^i} y_l^1 + \cdots + \overline{f^M} y_l^M}{\overline{f^i} + \cdots + \overline{f^M}} \quad (15)$$

$$y_{lr} = \frac{\underline{f^i} y_r^1 + \cdots + \underline{f^M} y_r^M}{\underline{f^i} + \cdots + \underline{f^M}} \quad (16)$$

$$y_{ur} = \frac{\overline{f^i} y_r^1 + \cdots + \overline{f^M} y_r^M}{\overline{f^i} + \cdots + \overline{f^M}} \quad (17)$$

With the above calculated inner bounds, the outer bounds ($\underline{y_l}$ and $\overline{y_r}$) can then be obtained using Equations 18 and 12 [32, 33], respectively as:

$$\underline{y_l} = \overline{y_l} - \left[ \frac{\sum_{i=1}^{m} (\overline{f^i} - \underline{f^i})}{\sum_{i=1}^{m} \overline{f^i} \times \sum_{i=1}^{m} \underline{f^i}} \times \frac{\sum_{i=1}^{m} \underline{f^i}(y_l^i - y_l^1) \times \sum_{i=1}^{m} \overline{f^i}(y_l^m - y_l^i)}{\sum_{i=1}^{m} \underline{f^i}(y_l^i - y_l^1) + \sum_{i=1}^{m} \overline{f^i}(y_l^m - y_l^i)} \right] \quad (18)$$

$$\overline{y_r} = \underline{y_r} + \left[ \frac{\sum_{i=1}^{m} (\overline{f^i} - \underline{f^i})}{\sum_{i=1}^{m} \overline{f^i} \times \sum_{i=1}^{m} \underline{f^i}} \times \frac{\sum_{i=1}^{m} \overline{f^i}(y_r^i - y_r^1) \times \sum_{i=1}^{m} \underline{f^i}(y_r^m - y_r^i)}{\sum_{i=1}^{m} \overline{f^i}(y_r^i - y_r^1) + \sum_{i=1}^{m} \underline{f^i}(y_r^m - y_r^i)} \right] \quad (19)$$

The lower and upper bounds of $y_l$ and $y_r$ can then be estimated using Equations 20 and 21.

$$y_l = \frac{\underline{y_l} + \overline{y_l}}{2} \quad (20)$$

$$y_r = \frac{\underline{y_r} + \overline{y_r}}{2} \quad (21)$$

Finally, a crisp output can be obtained by performing the defuzzification provided in Equation 22.

$$y = \frac{y_l + y_r}{2} \quad (22)$$

### 2.3. Robot Operating System (ROS)

ROS is an open-source message-based framework, that provides enhanced functionality for the development of large-scale service robots. Importantly, ROS provides hardware abstractions, low-level device controls, implementation of commonly used functionalities, message-passing, and package management for a set of connected robots and their subordinate components [30, 46].

Nodes, messages, topics, and services are the fundamental concepts of a ROS implementation [30]. The ROS master allows all ROS nodes to exchange messages between one another. Nodes publish messages

in topics; other nodes should subscribe to a topic in order to receive a message. This feature of ROS is advantageous enabling the reuse of publisher and subscriber implementations over multiple use cases. A ROS package can be directly used for organizing software components that provide easy-to-use functionality. A ROS package may contain ROS nodes, a ROS-independent library, a data-set, configuration files, third-party pieces of software, or anything else that logically constitutes a useful module.

ROS programs running on multiple computers can communicate over a network. This allows easy integration of software applications, packages, and drivers for robot kinematics visualization, data sharing, path planning, control, etc [47]. Further, ROS is multi-lingual, allowing users to develop different parts of the code with different languages including C++, Python, MATLAB, Java, etc [46].

### 2.4. Unified Model Language

First proposed by the *Object Management Group* (OMG) in 1997, *Unified Modeling Language* (UML) is a standard language which is basically used for specifying, visualizing, constructing, and documenting software systems using pictorial languages [48, 49]. UML provides multiple diagrams which are used for modeling a system in several levels of abstractions. UML diagrams are basically classified as behavioral and structural diagrams. Behavioral diagrams describe a system using activity, interaction, state flow, and use case diagrams. On the other hand, structural diagrams describe a system using class, composite structure, deployment, object, component, profile as well as package diagrams [49, 50].

UML by itself is not a programming language. However, it is highly used to conceptualize and structure computer codes using object-oriented design approaches. The object-oriented design uses objects as building blocks of a system. Objects may contain data, referred to as an attribute, and a logic sequence, which is referred to as an operation or method. The state and behavior of an object are defined and modeled within a class. This paper models FLC processes in an object-oriented approach using UML.

## 3. DEVELOPING FLC-ROS

In this section, we will discuss FLC ROS package's use cases, development architecture, configuration techniques as well as algorithms that enable MIMO FLC operations.

#### 3.0.1. FLS Library Use Case

The interactions between FLC ROS package, the controlled system, and the FLC application developers are illustrated using the use case diagram shown in Fig. 5.

In order to develop an FLC application using the developed FLC-ROS package, initially, users should configure the FLS package in accordance with the specification of the application that they have developed and build the ROS workspace. The developed FLC-ROS package performs fuzzification and inference as well as defuzzification operations for type-1 and interval type-2 FLCs. The controlled system may be a robot or any device for which the control inputs and outputs will be the inputs and outputs of the FLC, respectively.

### 3.1. FLC-ROS Configuration Structure

The developed FLC ROS package can be configured for MIMO fuzzy logic control applications using the structure presented in Fig. 6. Basic parameters of the FLC constitute the system's name, type, inference mechanism, number of inputs, number of outputs, rules, and so on. Each input and output has its own name and range as well as membership functions. Further, each membership function has its own linguistic parameter (name), range, and numeric parameters defining its shape. If the TSK inference technique is employed, rule consequent coefficients should be defined in the form of simple or type-1 coefficients. Rules are configured having linguistic antecedents, consequents, and inference logic, which could be combined using "$and$" or "$or$" operators.

Configuration is performed offline by using a cross-platform GUI or manually by editing a configuration file which is based on an extensible markup language (XML) format. The main window of the developed GUI is shown in Fig. 7. This main window enables the definition of basic parameters such as the FLC type,

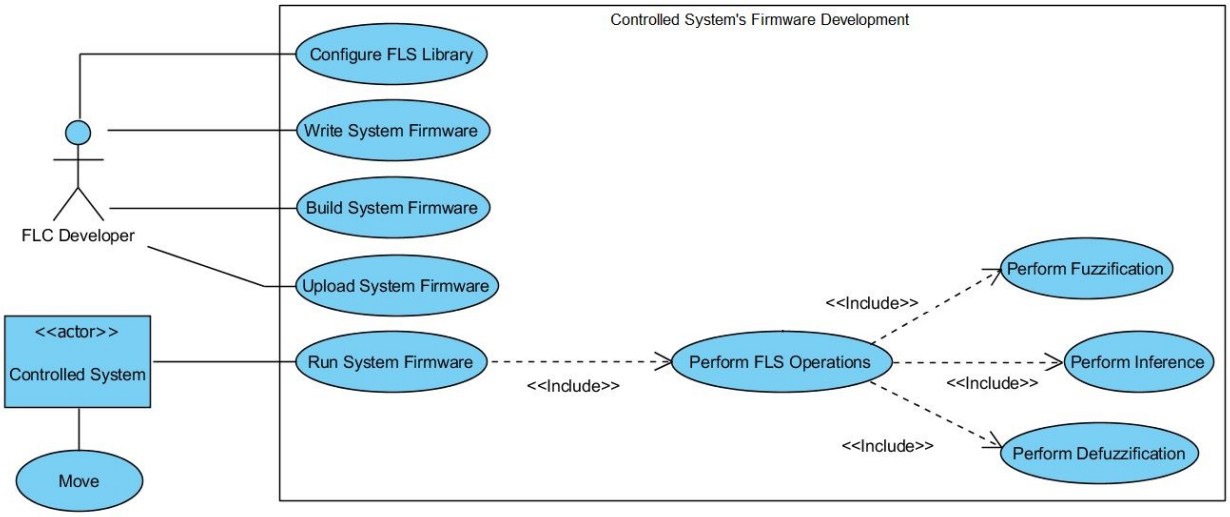



Fig. 5. FLC ROS package's use case diagram.

Table 1
Parameters of FLC ROS package MFs

| **Shape of MF** | **Triangular** | **Trapezoidal** | **Gaussian** | **Triangular-IT2** | **Trapezoidal-IT2** | **Gaussian-IT2** |
|---|---|---|---|---|---|---|
| **P1** | Left edge | Left edge | Mean | LMF left edge | LMF left edge | Mean-1 |
| **P2** | Center | Center left edge | Standard deviation | LMF center edge | LMF center left edge | Mean-2 |
| **P3** | Right edge | Center right edge | Unused | LMF right edge | LMF center right edge | Standard deviation |
| **P4** | Unused | Right edge | Unused | UMF left edge | LMF right edge | Unused |
| **P5** | Unused | Unused | Unused | UMF center edge | UMF left edge | Unused |
| **P6** | Unused | Unused | Unused | UMF right edge | UMF center left edge | Unused |
| **P7** | Unused | Unused | Unused | Unused | UMF center right edge | Unused |
| **P8** | Unused | Unused | Unused | Unused | UMF right edge | Unused |

inference method, aggregation method, and defuzzification method. Further, multiple GUI windows (not shown on the paper but are available on the publicly-made-available GitHub page) allow FLC developers to easily define or modify the input/output parameters and membership functions.

The membership function shapes that are supported by the developed ROS package are Triangular, Trapezoidal, and Gaussian. Each interval type-2 MF is defined using its upper and lower membership functions which could be Triangular, Trapezoidal, or Gaussian MFs. For configuration, the nth membership function is defined as $\texttt{MF}^{\texttt{n}} =< \texttt{Name} >, < \texttt{Shape} >, < \texttt{P1}, \texttt{P2}, ...\texttt{P8}, \texttt{Maximum} >$, where $\texttt{n}$ is ranging from $1, \cdots, k$, and $\texttt{P1} - \texttt{P8}$ are parameters used to define different edges and properties of type-1 and type-2 membership functions (See Table 1 for details).

The $\ell$th rule is structured as follows:

$$\texttt{R}^{\ell} =< \texttt{Antecedent} >, < \texttt{Consequent} >, < \texttt{Inference logic} >, < \texttt{TSKCoefficients} >$$

where $\ell$ is ranging from $1, \cdots, m$.

A saved configuration file can be uploaded to the developed FLC ROS package, enabling the reuse of design across multiple applications or implementations.

### *3.2. FLC ROS package Classes*

The FLC ROS package is developed using an object-oriented approach. In this case, with the purpose of making the package suitable for real-time control applications, FLC ROS is developed in C++. As

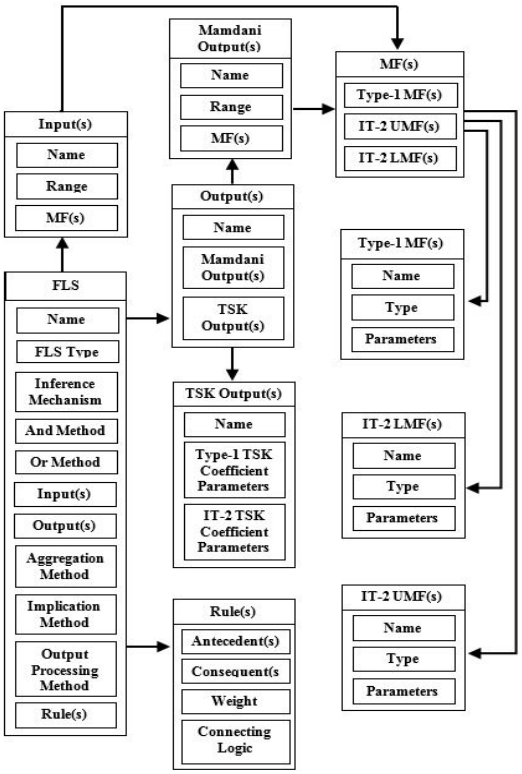



Fig. 6. Configuration structure of the developed FLC ROS package.

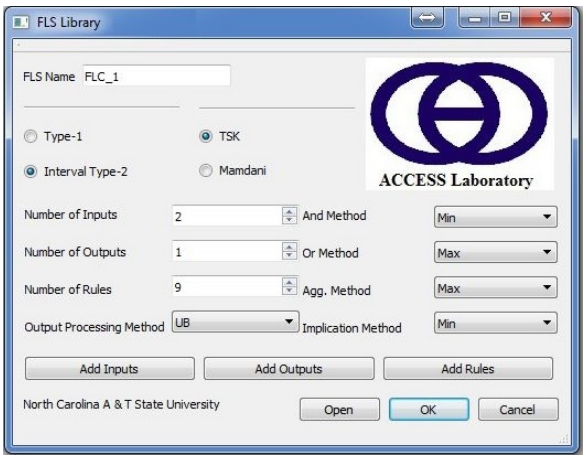



Fig. 7. The main window of the developed GUI

shown in Fig. 8, the developed ROS package has a parent class named `flslib`, which is composed of other classes including `fuzzify`, `inference`, and `defuzzify`. Class `flslib` has a method, named `perform_fls`, which is responsible for performing operations of fuzzification, inference, and defuzzification for both type-1 and IT-2 FLCs. This architecture allows FLC application developers to instantiate multiple objects having different configurations, facilitating the development of multiple FLCs in a single ROS application. Additionally, defining memberships, fuzzification, inference, and defuzzification as separate classes enable the development of customized FLC architectures through independent implementation and integration of fuzzy logic operations.

**Algorithm 1** Membership Value Calculation

**Input:** Crisp input, Input MF parameters
**Output:** Membership value
**Begin Procedure**

```
Read Crisp input and parameters
Calculate membership value
Return membership value
```

**End Procedure**

**Algorithm 2** Fuzzification

**Input:** Configuration, Crisp inputs
**Output:** Input fuzzy set(s)
**Begin Procedure**

```
Get crisp inputs;
Get configuration;
if FLC is type-1 then
    for all crisp inputs do
        for all MFs of the input do
            Identify the shape of the MF;
            Calculate the membership value;
            Store the fuzzified input;
        end for
    end for
else                                    ▷ FLC is IT-2
    for all crisp inputs do
        for all upper and lower IT-2 MFs of the input do
            Identify the shape of the MF;
            Calculate the membership value;
            Store the fuzzified input;
        end for
    end for
end if
Return the input fuzzy set(s);
```

**End Procedure**

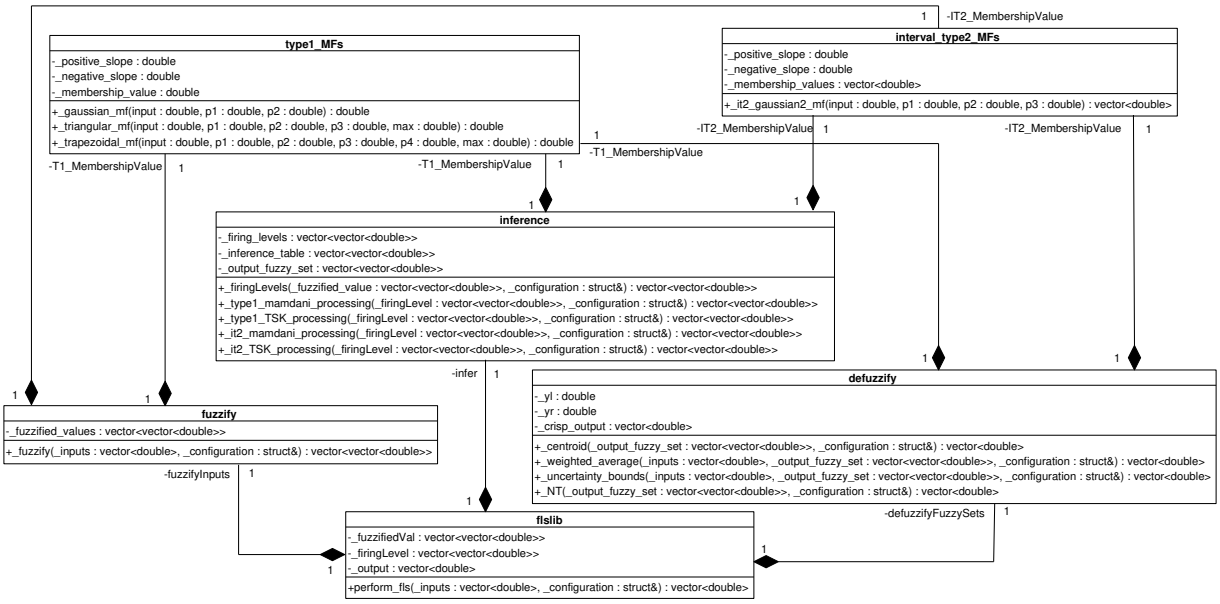



Fig. 8. UML Diagram of classes inside class *flslib*

### 3.2.1. *Fuzzification*

Fuzzification is a process to simply find the fuzzy values corresponding to each crisp input, which can be done using the membership functions. For this purpose, in the developed FLC ROS package, Class `flslib` contains the Class `fuzzify` that performs fuzzification operations. Class `fuzzify` is composed of other Classes for for different MFs including `type1_mfs` and `interval_type2_mfs`. Operations in these MF classes are used for calculating membership values using Algorithm 1. Having a system with multiple inputs and multiple outputs, Algorithm 2 in Class `fuzzify` performs MIMO fuzzification. This is achieved by an enumerative search over all inputs based on the shape of the corresponding membership function. For IT2 FLCs, the membership values will be calculated for both the upper and lower MFs.

### 3.2.2. *Inference*

Based on the configured rules, execution method of fuzzy logical operations and implication method, as well as the aggregation method, the inference process maps all input fuzzy sets to their respective output fuzzy sets. For each rule, the inference process initially calculates the associated firing levels. To make the output FSs suitable for the Mamdani defuzzification, the implication operation is performed for each rule output while the aggregation operation is performed for each FLC output. As shown in Fig. 8, the developed ROS package is composed of Class `inference`, capable of performing MIMO inference operations. This class is composed of other classes of type-1 and IT-2 MFs that are used for performing implication and aggregation operations. The developed algorithm for performing generic MIMO inference operations is described in Algorithm 3. By instantiating objects of Class `inference`, FLC application developers can perform the inference process independently, if they prefer to perform other processes (such as fuzzification and defuzzification) using a different method not supported by the developed ROS package.

### 3.2.3. *Output Processing*

The final step in an FLC process is output processing. The output processing of type-1 FLCs involves only defuzzification. However, the output processing of IT-2 FLCs involves type reduction followed by defuzzification. Class `defuzzify` is composed of operations which perform several kinds of type-reductions and defuzzifications, in accordance with the configuration set by the user. This enables application developers to customize FLC operations, if they prefer to per-

**Algorithm 3** Inference

**Input:** Configuration, crisp inputs, fuzzified inputs, rules
**Output:** Output fuzzy set(s)
**Begin Procedure**

```
Get the fuzzified input;
Get configuration;
if FLC type is type-1 then
    Compute the firing level of rules;
    if Inference is Mamdani then
        for all outputs do
            Perform implication on the consequent fuzzy sets;
            Perform aggregation to get the type-1 output fuzzy set;
        end for
    else                                   ▷ FLC is type-1 TSK
        for all outputs do
            Compute the TSK rule outputs and get the output fuzzy set;
        end for
    end if
else                                       ▷ FLC is IT-2
    Compute the firing level for the upper and lower bounds of the antecedent fuzzy sets of rules;
    if Inference is Mamdani then
        for all outputs do
            Perform implication on the upper and lower bounds of the consequent fuzzy sets;
            Perform aggregation on the upper and lower bounds of the consequent fuzzy sets to get the output fuzzy set;
        end for
    else                                   ▷ FLC is IT-2 TSK
        for all outputs do
            Compute the inner TSK uncertainty bounds (y_l and y_r);
            Compute the outer TSK uncertainty bounds (y_l and y_r);
            Compute the upper and lower bounds for the TSK output output fuzzy set (y_l and y_r);
        end for
    end if
end if
Return the output fuzzy set;
```

**End Procedure**

form other processes (fuzzification and inference) using methods not presently supported by the developed FLC ROS package.

In the developed ROS package, for all outputs, type-1 Mamdani FLCs are defuzzified using the centroid defuzzification technique while type-1 TSK FLCs are defuzzified using the weighted average defuzzification technique. Algorithm 4 implements the type-1 TSK FLC defuzzification. To obtain a crisp value of IT-2 Mamdani FLSs, for all outputs, the developed ROS package performs Nie-Tan type reduction on the output fuzzy set followed by an IT-2 defuzzification. If the configured FLC is IT-2 TSK FLS, uncertainty bounds output processing technique is used (See Sections 2.1 and 2.2 for details). As shown in Fig. 8, for assisting output processing of user-defined fuzzy sets, Class `defuzzify` is composed of other classes of type-1 and IT-2 MFs. Described in Algorithm 5 is the output processing for interval type-2 TSK FLCs.

**Algorithm 4** Type-1 TSK Defuzzification

**Input:** Configuration, crisp inputs, rule firing levels, rule output coefficients
**Output:** Crisp outputs
**Begin Procedure**

```
Read configuration, inputs, rule firing levels and rule output coefficients;
for all outputs do
    for all rules do
        Compute the corresponding rule output;
    end for
    Compute the crisp output using weighted average method;
    store the crisp output;
end for
Return crisp outputs;
```

**End Procedure**

### 3.2.4. Putting it all Together

The developed FLC ROS package performs FLC operations (fuzzification, inference, and defuzzification) for type-1 and IT-2 FLSs using the methodology presented in Algorithm 6. The source code of the developed package and relevant configuration examples are available on Github at *https://github.com/ACCESSLab/ros-fuzzy-library*.

**Algorithm 5** Interval type-2 TSK output Processing

**Input:** Configuration, crisp inputs, fuzzified upper and lower bound set, rule firing levels, rule output bounds
**Output:** Crisp outputs
**Begin Procedure**

*for* all outputs *do*
  Compute the inner uncertainty bounds ($\overline{y_l}$ and $\underline{y_r}$) using Equations (12) and (13);
  Compute the outer uncertainty bounds ($\underline{y_l}$ and $\overline{y_r}$) using Equations (18) and (19);
  Compute the upper and lower bounds for the output signal ($y_l$ and $y_r$) using Equations (20) and (21);
  Compute the crisp output $y$ using (22);
  Store the the crisp output $y$;
*end for*
Return crisp outputs;

**End Procedure**

**Algorithm 6** FLC ROS package Operations

**Input:** Crisp Inputs
**Output:** Crisp Outputs
**Begin Procedure**

Get configuration;
*while* perform_FLC = true *do*
  Get crisp inputs;
  Perform fuzzification;
  Perform inference;
  Perform defuzzification;
  Return crisp outputs;
*end while*

**End Procedure**

## 4. USE CASE: EMPLOYING THE DEVELOPED FLC PACKAGE FOR THE UAV ALTITUDE CONTROL

To evaluate the effectiveness of the developed ROS package, we configured the package to construct an IT-2 TSK fuzzy logic controller for the altitude control of a quadcopter UAV, simulated in Gazebo STIL simulation environment. The closed-loop controlled structure of the quadcopter UAV is based on Fig. 9.

The IT-2 TSK FLC takes two inputs and produces one control output. The inputs are the altitude error, $Z_{err}$, and its derivative, $\dot{Z}_{err}$, whereas the control output is the throttle level. In general, the membership functions and rule-base of FLC can be designed and tuned experimentally based on the prior knowledge about the system and conducting a trial-and-error approach, or they can be found via different techniques based on optimization (e.g., genetic algorithm [51], simulated annealing [52], or tabu search [53]), learning [54], or clustering [55]. Either way, the developed ROS package can be employed for the implementation of the designed FLC regardless of the design approach. In this case-study, we designed the membership functions and rule-base of the FLC experimentally and selected the input IT-2 MFs of the TSK FLC as shown in Figures 10(a) and 10(b). The rule-base of the FLC is stated in Table 2. For example,

$$R^{\ell}\text{: } \textbf{\textit{IF}}\ Z_{err}\ \textit{is}\ L\ \textit{and}\ \dot{Z}_{err}\ \textit{is}\ 0 \quad \textbf{\textit{THEN}}\ y\ \text{is}\ ML \tag{23}$$

The output uncertainties of the first-order TSK FLC coefficients ($Z_{err}$ and $\dot{Z}_{err}$) are stated in Table 3. For example, for the rule in Equation 24, the output will be:

$$y = c_1 Z_{err} + c_2 \dot{Z}_{err} \tag{24}$$

where the upper and lower bounds of coefficients $c_1$ and $c_2$ are given in the third row of Table 3.

The output processing unit for developing the FLC uses uncertainty bounds aggregation method, described in Section 2.2. Presented in Fig. 11 is the UAV FLC development process in ROS using the developed package described in Section 3. Initially, the FLC parameters are configured using the provided GUI. Following this, an object with the configured properties is instantiated to perform UAV FLC operations in ROS. Fig. 12 shows the sequence diagram which describes how FLC ROS package operations interact for performing fuzzification, inference, and defuzification operations at every control time-step.

The control simulation results are shown in Figures 13(a) and 13(b). These results are also compared with that of a classical PD controller. For making the comparison fair, the gains of the PD controller were set at the centers of the output uncertainty of the implemented IT-2 TSK FLC as follows:

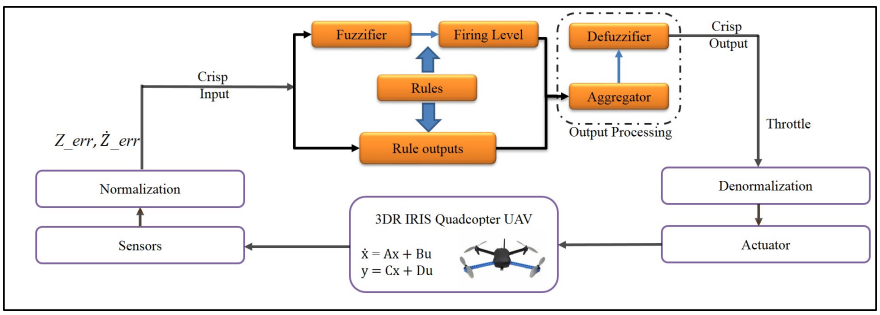
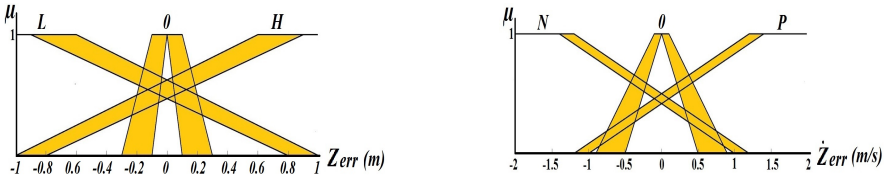



Fig. 9. Interval type-2 TSK UAV altitude control structure.




(a) (b)

Fig. 10. (a) Membership functions of $Z_{err}$, (b) Membership functions of $\dot{Z}_{err}$.

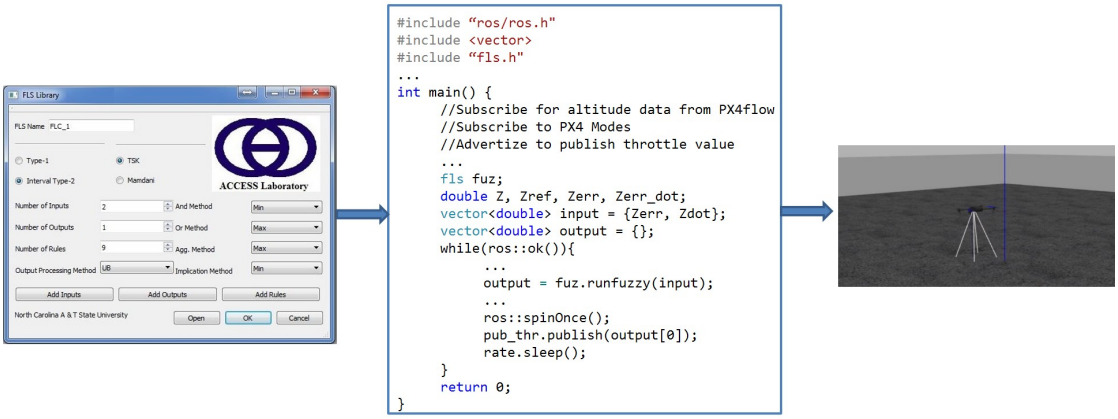



Fig. 11. Example steps for UAV control using FLC ROS package and Gazebo SITL simulator.

Table 2

Rule Base for the IT-2 TSK FLC for the attitude control of the UAV

| | | $Z_{err}$ | | |
|---|---|---|---|---|
| | | **L** | **0** | **H** |
| $\dot{Z}_{err}$ | **N** | L | ML | MS |
| | **0** | ML | S | ML |
| | **P** | MS | ML | L |

$$y = 0.55Z_{err} + 0.45\dot{Z}_{err} \tag{25}$$

From the simulation results, it can be observed that the interval type-2 TSK FLC has successfully controlled the altitude of the UAV with a smaller overshoot and faster settling time compared to the classical PD controller.

Table 3

Rule coefficient bounds

| **Rule Number** | **Rule Output Coefficient Label** | $\underline{c_1}$ | $\overline{c_1}$ | $\underline{c_2}$ | $\overline{c_2}$ |
|---|---|---|---|---|---|
| 1 | S | 0.4 | 0.45 | 0.3 | 0.35 |
| 2 | MS | 0.5 | 0.6 | 0.3 | 0.35 |
| 3 | ML | 0.5 | 0.6 | 0.45 | 0.5 |
| 4 | L | 0.65 | 0.7 | 0.55 | 0.6 |

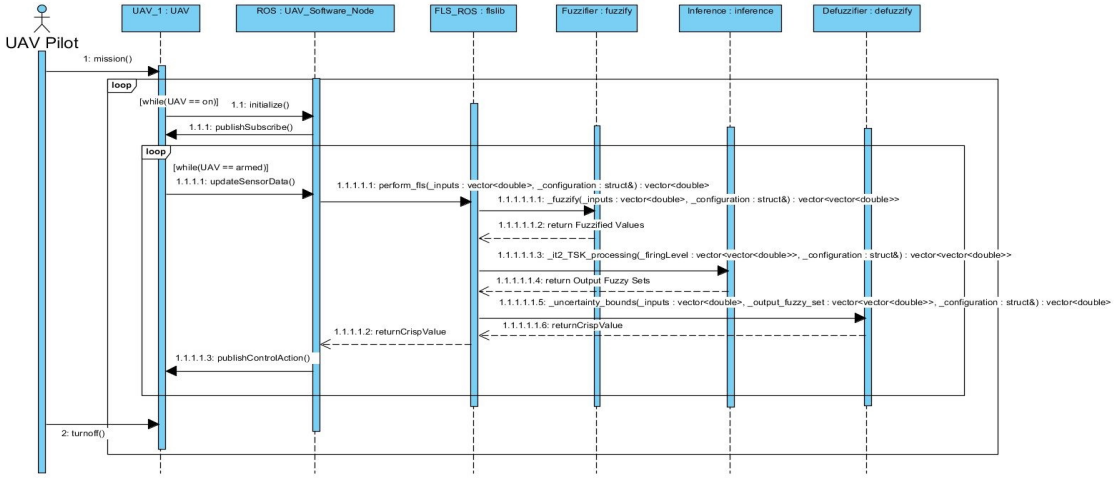
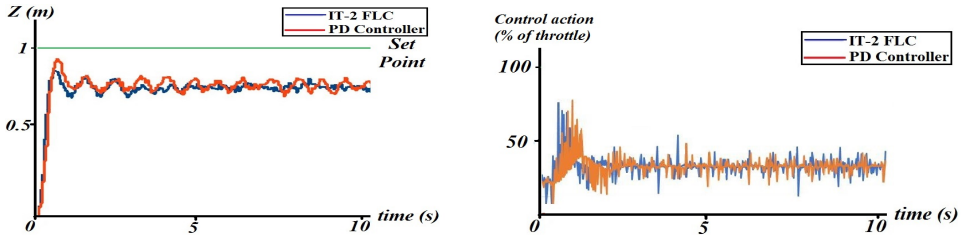



Fig. 12. UML sequence diagram of FLC execution using the developed FLC ROS package.




(a) (b)

Fig. 13. (a) UAV altitude using IT-2 TSK and PD controllers, (b) UAV control signal using IT-2 TSK and PD controllers.

## 5. CONCLUSION

This paper developed a reconfigurable FLC ROS package using an object-oriented approach and UML. For this purpose, we developed an implementation architecture, configuration technique, and algorithms that enable easier implementation of generic and configurable MIMO FLCs for ROS control applications. The software architectures and algorithms of the developed FLC ROS package were realized using C++ programming language. Further, possible use cases of the FLC ROS package for developing MIMO type-1 and interval type-2 were explained. The developed FLC ROS package allows the user to configure the control structure to use Mamdani or TSK fuzzy inference mechanisms. Further, to enhance the computation cost for real-time control applications, the developed FLC ROS package allows for adopting the uncertainty bounds and Nie-Tan output processing techniques. A user-friendly graphical user interface was developed to facilitate the configuration of MIMO fuzzy logic control parameters such as number of inputs, number of outputs, rules, membership functions, inference methods, type reduction, and defuzzification. The effectiveness of the developed package was demonstrated by applying to the altitude control of a quadcopter UAV in Gazebo simulation environment. The simulation results were then compared with a PD controller. The results have shown that the developed ROS package can actually assist in realizing ROS-based fuzzy logic control applications. Inspired by the work in [56, 57], one of the research directions that we will pursue is to enhance the developed tool for the implementation of adaptive and fault tolerant fuzzy control systems. Our focus in this paper was on type-1 and interval type-2 FLCs with Mamdani or TSK inferences. Following the same procedure, the developed tool will be extended to other FLC types and inferences.

## Acknowledgment

This work is supported by National Science Foundation under the award number 1832110 and 2000320 as well as the Air Force Research Laboratory and Office of the Secretary of Defense under agreement number FA8750-15-2-0116.

## References

[1] E. Wolfgang, Introduction to artificial intelligence, Springer, Cham, 2019. doi:10.1007/978-3-319-58487-4.

[2] M. Rodd, H. Verbruggen, A. Krijgsman, Artificial intelligence in real-time control, Engineering Applications of Artificial Intelligence 5 (5) (1992) 385–399. doi:https://doi.org/10.1016/0952-1976(92)90010-H.

[3] J. L. Castro, Fuzzy logic controllers are universal approximators, IEEE Transactions on Systems, Man, and Cybernetics 25 (4) (1995) 629–635. doi:10.1109/21.370193.

[4] S. Ramyar, A. Homaifar, A. Anzagira, A. Karimoddini, S. Amsalu, A. Kurt, Fuzzy modeling of drivers' actions at intersections, in: 2016 World Automation Congress (WAC), 2016, pp. 1–6. doi:10.1109/WAC.2016.7582966.

[5] L. A. Zadeh, Outline of a New Approach to the Analysis of Complex Systems and Decision Processes, IEEE Transactions on Systems, Man, and Cybernetics SMC-3 (1) (1973) 28–44. doi:10.1109/TSMC.1973.5408575.

[6] D. Dubois, H. Prade, L. Ughetto, Fuzzy Logic, Control Engineering and Artificial Intelligence, Springer Netherlands, Dordrecht, 1999, pp. 17–57. doi:10.1007/978-94-011-4405-6_2.

[7] L. Zadeh, The concept of a linguistic variable and its application to approximate reasoning—II, Information Sciences 8 (4) (1975) 301–357. doi:10.1016/0020-0255(75)90046-8.

[8] J. Mendel, R. John, Type-2 fuzzy sets made simple, IEEE Transactions on Fuzzy Systems 10 (2) (2002) 117–127. doi:10.1109/91.995115.

[9] J. M. Mendel, R. I. John, F. Liu, Interval type-2 fuzzy logic systems made simple, IEEE Transactions on Fuzzy Systems 14 (6) (2006) 808–821. doi:10.1109/TFUZZ.2006.879986.

[10] A. Hailemichael, S. M. Salaken, A. Karimoddini, A. Homaifar, K. Abbas, S. Nahavandi, Developing a computationally effective interval type-2 tsk fuzzy logic controller, Journal of Intelligent & Fuzzy Systems 38 (2) (2020) 1915–1928. doi:10.3233/JIFS-190446.

[11] M. Biglarbegian, W. W. Melek, J. M. Mendel, Design of novel interval type-2 fuzzy controllers for modular and reconfigurable robots: Theory and experiments, IEEE Transactions on Industrial Electronics 58 (4) (2011) 1371–1384. doi:10.1109/TIE.2010.2049718.

[12] D. Wu, W. Tan, A type-2 fuzzy logic controller for the liquid-level process, in: IEEE International Conference on Fuzzy Systems, Vol. 2, 2004, pp. 953–958 vol.2. doi:10.1109/FUZZY.2004.1375536.

[13] A. Hailemichael, M. Behniapoor, A. Karimoddini, Development of an Interval Type-2 TSK Fuzzy Logic Attitude Controller for a UAV, in: 2018 International Conference on Unmanned Aircraft Systems (ICUAS), 2018, pp. 1003–1009. doi:10.1109/ICUAS.2018.8453330.

[14] C. Wen, Z. Hui, L. Tao, L. Yuling, Intelligent traffic signal controller based on type-2 fuzzy logic and nsgaii, Journal of Intelligent & Fuzzy Systems 29 (6) (2015) 2611–2618. doi:10.3233/IFS-151964.

[15] B. Safarinejadian, B. Bagheri, P. Ghane, Fault detection in nonlinear systems based on type-2 fuzzy sets and bat optimization algorithm, Journal of Intelligent & Fuzzy Systems 28 (1) (2015) 179–187. doi:10.3233/IFS-141288.

[16] Z. Wang, S. Ramyar, S. M. Salaken, A. Homaifar, S. Nahavandi, A. Karimoddini, A collision avoidance system with fuzzy danger level detection, in: 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 283–288. doi:10.1109/IVS.2017.7995733.

[17] A. Davari, M. Hamiruce Marhaban, S. Bahari Mohd Noor, M. Karimadini, A. Karimoddini, Parameter estimation of k-distributed sea clutter based on fuzzy inference and gustafson–kessel clustering, Fuzzy Sets and Systems 163 (1) (2011) 45–53, theme: Classification and Modelling. doi:10.1016/j.fss.2010.09.008.

[18] S. Ramyar, M. G. Sefidmazgi, S. Amsalu, A. Anzagira, A. Homaifar, A. Karimoddini, A. Kurt, Modeling driver behavior at intersections with takagi-sugeno fuzzy models, in: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, 2015, pp. 2378–2383. doi:10.1109/ITSC.2015.384.

[19] A. Homaifar, A. Karimoddini, M. Heiges, M. A. Khan, B. A. Erol, S. Nazmi, A software tool for evaluating unmanned autonomous systems, The ITEA Journal of Test and Evaluation 41 (3) (2020) 188–195.

[20] A. Karimoddini, S. Grebreyohannes, A. Homaifar, Automatic testing tool for testing autonomous systems, US Patent App. 17/148,909 (Aug. 5 2021).

[21] P. Ejegwa, S. Wen, Y. Feng, W. Zhang, N. Tang, Novel pythagorean fuzzy correlation measures via pythagorean fuzzy deviation, variance and covariance with applications to pattern recognition and career placement, IEEE Transactions on Fuzzy Systems (2021). doi:10.1109/TFUZZ.2021.3063794.

[22] S. Gebreyohannes, A. Karimoddini, A. Homaifar, A. Esterline, Formal verification of a fuzzy rule-based classifier using the prototype verification system, in: G. A. Barreto, R. Coelho (Eds.), Fuzzy Information Processing, Springer International Publishing, Cham, 2018, pp. 1–12. doi:10.1007/978-3-319-95312-0_1.

[23] S. Greenfield, Type-2 fuzzy logic: Circumventing the defuzzi-

fication bottleneck, De Montfort University, PhD dissertation, 2012.

[24] J. Boren, S. Cousins, Exponential Growth of ROS, IEEE Robotics Automation Magazine 18 (1) (2011) 19–20. doi:10.1109/MRA.2010.940147.

[25] P. Estefo, J. Simmonds, R. Robbes, J. Fabry, The robot operating system: Package reuse and community dynamics, Journal of Systems and Software 151 (2019) 226–242. doi:10.1016/j.jss.2019.02.024.

[26] L. Garber, Robot os: A new day for robot design, Computer 46 (12) (2013) 16–20. doi:10.1109/MC.2013.434.

[27] L. Peppoloni, F. Brizzi, C. A. Avizzano, E. Ruffaldi, Immersive ROS-integrated framework for robot teleoperation, in: 2015 IEEE Symposium on 3D User Interfaces (3DUI), 2015, pp. 177–178. doi:10.1109/3DUI.2015.7131758.

[28] D. G. Schneider, L. Lima da Silva, P. Diehl, A. H. R. Leite, G. S. Bastos, Robot Navigation by Gesture Recognition with ROS and Kinect, in: 2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR), 2015, pp. 145–150. doi:10.1109/LARS-SBR.2015.21.

[29] Y. Chen, X. Wang, S. Hong, X. Zhong, C. Zou, Motion planning implemented in ROS for mobile robot, in: 2017 29th Chinese Control And Decision Conference (CCDC), 2017, pp. 7149–7154. doi:10.1109/CCDC.2017.7978473.

[30] A. Koubaa, Robot Operating System (ROS) The Complete Reference (Volume 2), Springer, 2017. doi:10.1007/978-3-319-54927-9.

[31] M. Costa de Oliveira, M. Rocha Facury, Writing fuzzy rules directly in a C++ source code, in: Proceedings of IEEE 5th International Fuzzy Systems, Vol. 1, 1996, pp. 522–528 vol.1. doi:10.1109/FUZZY.1996.551795.

[32] H. Wu, J. Mendel, Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems, IEEE Transactions on Fuzzy Systems 10 (5) (2002) 622–639. doi:10.1109/TFUZZ.2002.803496.

[33] N. Enyinna, A. Karimoddini, D. Opoku, A. Homaifar, S. Arnold, Developing an interval type-2 tsk fuzzy logic controller, in: 2015 Annual Conference of the North American Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing (WConSC), 2015, pp. 1–6. doi:10.1109/NAFIPS-WConSC.2015.7284160.

[34] M. Nie, W. W. Tan, Towards an efficient type-reduction method for interval type-2 fuzzy logic systems, in: 2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence), 2008, pp. 1425–1432. doi:10.1109/FUZZY.2008.4630559.

[35] E. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, International Journal of Man-Machine Studies 7 (1) (1975) 1–13. doi:https://doi.org/10.1016/S0020-7373(75)80002-2.

[36] P. Martin Larsen, Industrial applications of fuzzy logic control, Vol. 12, 1980. doi:https://doi.org/10.1016/S0020-7373(80)80050-2.

[37] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Transactions on Systems, Man, and Cybernetics SMC-15 (1) (1985) 116–132. doi:10.1109/TSMC.1985.6313399.

[38] J. M. Mendel, Uncertain rule-based fuzzy logic system: introduction and new directions, Prentice–Hall PTR, 2001. doi:10.1007/978-3-319-51370-6.

[39] N. Karnik, J. Mendel, Applications of type-2 fuzzy logic systems: handling the uncertainty associated with surveys 3 (1999) 1546–1551. doi:10.1109/FUZZY.1999.790134.

[40] J. M. Mendel, General type-2 fuzzy logic systems made simple: A tutorial, IEEE Transactions on Fuzzy Systems 22 (5) (2014) 1162–1182. doi:10.1109/TFUZZ.2013.2286414.

[41] H. Hagras, Type-2 FLCs: A New Generation of Fuzzy Controllers, IEEE Computational Intelligence Magazine 2 (1) (2007) 30–43. doi:10.1109/MCI.2007.357192.

[42] J. M. Mendel, F. Liu, D. Zhai, $\alpha$-plane representation for type-2 fuzzy sets: Theory and applications, IEEE Transactions on Fuzzy Systems 17 (5) (2009) 1189–1207. doi:10.1109/TFUZZ.2009.2024411.

[43] H. Hamrawi, S. Coupland, R. John, Type-2 Fuzzy Alpha-Cuts, IEEE Transactions on Fuzzy Systems 25 (3) (2017) 682–692. doi:10.1109/TFUZZ.2016.2574914.

[44] D. Wu, On the Fundamental Differences Between Interval Type-2 and Type-1 Fuzzy Logic Controllers, IEEE Transactions on Fuzzy Systems 20 (5) (2012) 832–848. doi:10.1109/TFUZZ.2012.2186818.

[45] J. Li, R. John, S. Coupland, G. Kendall, On nie-tan operator and type-reduction of interval type-2 fuzzy sets, IEEE Transactions on Fuzzy Systems 26 (2) (2018) 1036–1039. doi:10.1109/TFUZZ.2017.2666842.

[46] J. Kerr, K. Nickels, Robot operating systems: Bridging the gap between human and robot, in: Proceedings of the 2012 44th Southeastern Symposium on System Theory (SSST), 2012, pp. 99–104. doi:10.1109/SSST.2012.6195127.

[47] B. Dieber, S. Kacianka, S. Rass, P. Schartner, Application-level security for ROS-based applications, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4477–4482. doi:10.1109/IROS.2016.7759659.

[48] G. B. J. Rumbaugh, I. Jacobson, The unified modeling language user guide, Addison Wesley (1998).

[49] N. G. N. D. Muhazam Mustapha, UML diagram for design patterns, ICSECS (2011). doi:10.1007/978-3-642-22203-0_19.

[50] M. N. Alanazi, Basic rules to build correct uml diagrams, in: 2009 International Conference on New Trends in Information and Service Science, 2009, pp. 72–76. doi:10.1109/NISS.2009.252.

[51] Y.-S. Zhou, L.-Y. Lai, Optimal design for fuzzy controllers by genetic algorithms, IEEE Transactions on Industry Applications 36 (1) (2000) 93–97. doi:10.1109/28.821802.

[52] J. Garibaldi, E. Ifeachor, Application of simulated annealing fuzzy model tuning to umbilical cord acid-base interpretation, IEEE Transactions on Fuzzy Systems 7 (1) (1999) 72–84. doi:10.1109/91.746314.

[53] A. Bağiş, Determining fuzzy membership functions with tabu search—an application to control, Fuzzy Sets and Systems 139 (1) (2003) 209–225. doi:https://doi.org/10.1016/S0165-0114(02)00502-X.

[54] M. J. Er, C. Deng, Online tuning of fuzzy inference systems using dynamic fuzzy q-learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34 (3) (2004) 1478–1489. doi:10.1109/TSMCB.2004.825938.

[55] A. Karimoddini, K. Salahshoor, A. Fatehi, M. Karimadini, A new approach for online fuzzy identification by potential clustering including rule reduction, in: 2007 European Control Conference (ECC), 2007, pp. 747–754. doi:10.23919/ECC.2007.7068445.

[56] K. Sun, H. R. Karimi, J. Qiu, Finite-time fuzzy adaptive quantized output feedback control of triangular structural systems, Information Sciences 557 (2021) 153–169. doi:https://doi.org/10.1016/j.ins.2020.12.059.

[57] K. Sun, L. Liu, J. Qiu, G. Feng, Fuzzy adaptive finite-time fault-tolerant control for strict-feedback nonlinear systems, IEEE Transactions on Fuzzy Systems 29 (4) (2021) 786–796. doi:10.1109/TFUZZ.2020.2965890.