Development of a Generic and Configurable Fuzzy Logic Systems Library for Real-Time Control Applications using an Object-oriented Approach

Abel Hailemichael, Solomon Gebreyohannes, Ali Karimoddini *, Kaushik Roy, Abdollah Homaifar
North Carolina A&T State University
Greensboro, NC, USA

Abstract—Since fuzzy logic controllers (FLCs) can handle complex systems without knowing much about the systems' mathematical model, they are widely used for a range of robotic control applications. Further, the ability of FLCs (particularly, type-2 FLCs) to effectively capture and accommodate uncertainties has made them one of the suitable choices for implementing robotic control applications in uncertain environments. However, developing type-1 and type-2 FLCs for real-time robotic control applications is relatively more challenging than developing traditional controllers such as PID controllers. The reason is, the fuzzy logic calculations involved are more complex and not much tools have been developed to assist FLC application developers. In this paper, therefore, using an object-oriented approach and unified model language (UML), we demonstrate a systematic approach for developing a new generic and configurable fuzzy logic system (FLS) library that eases the implementation of real-time type-1 and interval type-2 FLC applications based on both Mamdani and Takagi-Sugeno-Kang (TSK) inference mechanisms. To evaluate the developed library, we have implemented it for the interval type-2 TSK fuzzy logic altitude control of a quadcopter unmanned aerial vehicle (UAV). The response of this fuzzy logic controller is then compared with the response of a classical PD controller.

Index Terms—Robot control, Type-1 Fuzzy Logic System, Interval Type-2 Fuzzy Logic System, TSK, Mamdani, Object-oriented Design, UML.

I. INTRODUCTION

First advanced by Zadeh [1], a fuzzy logic system (FLS) is an approach for computing based on human-like expressions and degrees of truth. FLSs use membership functions (MFs) to define the degree of truth for linguistic variables. By using membership functions and rule based inference mechanisms, FLSs are exceptionally powerful in mimicking human decision-making. Furthermore, along with their ability to express the behavior of a complex system without knowing much about its mathematical model, FLSs (particularly type-2 FLSs) are capable of effectively capturing and handling uncertainties [2], [3]. This has made the use of type-2 FLSs a suitable choice for the development of real-time robot control and decision making applications in uncertain environments [4]–[7].

Nevertheless, the development process of fuzzy logic controllers (FLCs) is relatively more complex than commonly-used controllers such as PID controllers. On the other hand, there does not exist a user-friendly development tools that eases the implementation of all types of FLCs. This is a challenge for the wide use of FLCs, particularly type-2 FLCs, in real-world applications. Though some work has been done to develop an FLS library for the implementation of FLCs [8]–[10], they are limited to a particular class of FLCs, do not support the development of interval type-2 (IT-2) Mamdani FLCs, and do not support the development of first-order Takagi-Sugeno-Kang (TSK) FLCs. Therefore, the challenge for finding a flexible tool which provides the required functionality for the design and implementation of FLCs still remains open. To ease this challenge, this paper provides a systematic approach for implementing a generic and configurable fuzzy logic system library that can be used for the development of real-time type-1 and interval type-2 FLCs, based on both Mamdani [11] and TSK [12] inference methodologies. For allowing FLC developers to implement several types of FLCs without demanding them the detailed mathematical knowledge of type-1 and interval type-2 (IT-2) FLSs and for increasing reuse of design, the library is developed using an object-oriented design approach.

Thus, the contributions of this paper are the systematic modeling of a software architecture that enables the implementation of a generic and configurable FLS library and the realization of an FLS library that can be used for the implementation of real-time type-1 and interval type-2 FLCs, particularly for robotic applications. The developed FLS library is modular and eases the implementation of type-1 Mamdani, type-1 TSK, interval type-2 Mamdani as well as interval type-2 TSK FLSs. Unlike previously developed libraries, it supports the easy implementation of zero-order and first-order TSK FLSs.

The structure of the code and the interactions between the library user, the robot hardware as well as the robot firmware are formulated and illustrated using UML diagrams [13]–[15]. To enable effective real-time implementations, the library is developed using C++ programming language.
For making the library modular, this paper presents the fuzzy logic system as a single class, that encloses other classes of membership functions, fuzzification, inference and defuzzification. Furthermore, for enabling the development of a computationally less expensive type-2 FLCs, this FLS library is developed in such a way that it supports the implementation of interval type-2 FLSs that use uncertainty bounds aggregation methodology [2], [16] and Nie-Tan [17] type-reduction technique. Additionally, to make the library user-friendly, a cross-platform graphical user interface (GUI) tool is developed allowing the FLC developers to configure system parameters such as the FLS type, inference mechanism, number of inputs, number of outputs, membership functions, rules, implication (clipping) method, aggregation method, defuzzification method and so on. The system parameters can also be set using a configuration file which is developed specifically for the FLS library.

For evaluating the effectiveness of the developed library, an IT-2 TSK fuzzy logic altitude control of a quadcopter unmanned aerial vehicle (UAV) is simulated using the developed library in Gazebo software in the loop (STIL) simulation environment. The response of the interval type-2 TSK fuzzy logic altitude controller is compared with the response of a classical PD controller.

The rest of this paper is organized as follows. In Section II, we briefly discuss the necessary preliminaries and backgrounds in fuzzy control systems. In section III, we discuss the developed FLS library’s organization and functionality. Section IV describes the simulation results for the altitude control of a quadcopter UAV using the developed library. The paper is concluded in Section V.

II. PRELIMINARIES

Type-1 and type-2 fuzzy sets can be defined as follows.

**Definition 1: Type-1 Fuzzy Set**

A type-1 fuzzy set is composed of fuzzy set elements and their corresponding membership values \((x, \mu_A(x))\), \(\mu_A(x)\in[0,1]\). A fuzzy set can be formally defined as:

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

where, \(X\) is the universe of discourse, \(A\) is the fuzzy set and \(\sum\) is the collection of elements of the set.

**Definition 2: Type-2 Fuzzy Set**

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

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

An interval type-2 fuzzy set is a type-2 fuzzy set in which the secondary grade values are always unity. Type-2 and interval type-2 fuzzy sets are capable of capturing uncertainties in degree of memberships.

The development process of both type-1 and type-2 FLSs involves the processes of fuzzification, inference and defuzzification. By using MFs, the process of fuzzification assigns membership value to crisp inputs to obtain the input fuzzy set. The inference process maps input fuzzy sets to output fuzzy sets according to the defined rules. By performing defuzzification, a crisp output is obtained from the output fuzzy sets. While processing type-2 FLSs, the membership functions as well as the output fuzzy sets of the inference engine are type-2. Therefore, for converting the calculated type-2 fuzzy sets to a type-1 fuzzy sets, a type-reduction unit is required prior to defuzzification.

Type-2 FLSs are computationally expensive. Employing IT-2 fuzzy sets significantly reduce the computation costs of type-2 FLSs while maintaining their major advantages [18], [19].

Fuzzy rules describe the relation between inputs and outputs of a system. A \(p\) inputs and \(q\) outputs Mamdani FLS rule can be defined as,

\[
R^\ell: IF \ x_1 \ is \ F_1^\ell, \ and \ x_2 \ is \ F_2^\ell \ and \ldots \ and \ x_p \ is \ F_p^\ell \ THEN \ y_1 \ is \ G_1^\ell, \ y_2 \ is \ G_2^\ell, \ldots \ y_q \ is \ G_q^\ell
\]

where \(R^\ell\) is the \(\ell\)th rule, \(F_p^\ell\) is the activated antecedent fuzzy set for input channel \(x_p\), and \(G_q^\ell\) is the activated consequent fuzzy set for output channel \(y_q\).

The firing level for the \(\ell_{th}\) rule is the \(t\)-norm of the activated antecedent fuzzy sets as defined by the rules. For obtaining the final crisp outputs of a type-1 Mamdani FLS, centroid defuzzification method can be used [2]. Similarly, for obtaining the final crisp outputs of IT-2 Mamdani FLSs, Nie-Tan defuzzification method can be used [17].

A \(p\) inputs and \(q\) outputs TSK FLS rule can be defined as,

\[
R^\ell: IF \ x_1 \ is \ F_1^\ell, \ and \ x_2 \ is \ F_2^\ell \ and \ldots \ and \ x_p \ is \ F_p^\ell \ THEN \ y_1 = c_0,1 + c_1^\ell x_1 + \ldots + c_p,1 x_p, \ y_2 = c_0,2 + c_1^\ell x_1 + \ldots + c_p,2 x_p, \ldots \ y_q = c_0,q + c_1^\ell x_1 + \ldots + c_p,q x_p
\]

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 \(c_0,1, c_1,1 \ldots c_p,q\) are TSK output coefficients for outputs \(y_1, y_2, \ldots, y_q\). Weighted average defuzzification method may be used for obtaining the final crisp output of type-1 TSK FLSs. Similarly, uncertainty bounds output processing technique [16] may be used for obtaining the final crisp output of IT-2 FLSs.

III. DEVELOPING THE PROPOSED FLS LIBRARY

A. FLS Library Use Case

The interaction between the FLS library, the controlled system and the FLC application developers is illustrated on the use case diagram shown in Fig. 1.

To get the required functionality from the controlled system, the FLC application developers have to configure the
The FLS library in accordance with the specification of the application that they have developed. To perform FLS operations, the controlled system’s firmware needs to include the FLS library. The Perform FLS Operations use case includes performing fuzzification, inference as well as defuzzification. The controlled system may be a robot or any device which is expected to be controlled using an FLC. If we assume the controlled system to be a robot, one example use case for it would be to move.

B. FLS Library Configuration Structure

The configured properties of the developed FLS library are structured as shown in Fig. 2. The configuration is set offline by using the 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 GUI shown in Fig. 3. Different windows of the GUI allow FLC developers to easily define properties of a multi-input multi-output FLS. A saved configuration file can be uploaded to the developed FLS library, enabling reuse of design across different implementations.

C. FLS Library Classes

The FLS library is developed using an object-oriented approach and implemented in C++. However, the architecture may be reused across different implementations. As shown in Fig. 4, the developed FLS library has a top class named flslib, which is composed of other classes of fuzzify, inference and defuzzify. This class has a method named, perform_fls, which is used for performing fuzzification, inference and defuzzification operations for both type-1 and IT-2 FLSs. This allows FLC application developers to instantiate different objects with different configurations, enabling them to have multiple FLSs in their application. Additionally, defining MFs, fuzzification, inference and defuzzification as separate classes with their own attributes and operations allows FLC application developers to perform any of these FLS operations independently. In the developed library, the crisp inputs and crisp outputs of the FLS are represented as elements of a vector. Hence, all crisp inputs are passed to the FLS library operations as a vector and all crisp outputs from the FLS library operations are returned as a vector. The
Algorithm 1 Fuzzification

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

Begin Procedure
1: Get crisp inputs;
2: Get configuration;
3: if FLS is type-1 then
  4: for all crisp inputs do
    5: for all MFs of the input do
      6: Identify the shape of the MF;
      7: Calculate the membership value;
      8: Store the fuzzified input;
    else 
      9: # FLS is IT-2
        10: for all crisp inputs do
          11: for all upper and lower IT-2 MFs of the input do
            12: Identify the shape of the MF;
            13: Calculate the membership value;
            14: Store the fuzzified input;
        15: Return the input fuzzy set(s);

End Procedure

organization of these classes as well as the relation between them is shown in Fig. 4.

1) Fuzzification: In the developed FLS library, class flslib is composed of class fuzzify that performs fuzzification operations. Class fuzzify is composed of other classes of MFs: type1_mfs and interval_type2_mfs. Operations in these MF classes are used for calculating membership values. The algorithm implemented in class fuzzify for performing fuzzification is presented in Algorithm 1.

2) Inference: Based on the configured rules, fuzzy logical operations execution method, implication method as well as aggregation method, the inference process maps all input FSs to an output FSs. For mapping the input FSs to an output FSs, for each rule, the inference process initially calculates the firing level. For Mamdani FLSs, to make the output FSs suitable for defuzzification, the implication process is performed for each rule output while the aggregation operation is performed for each FLS output. In the developed FLS library, for performing the fuzzy inference operation, class flslib is composed of class inference. As shown in Fig. 4, class inference composed of other classes of type-1 and IT-2 MFs which are used for implication and aggregation of Mamdani FLSs. The process of implication is not required for TSK FLSs. The algorithm which is used by class inference for the inference operation is described in Algorithm 2.

3) Defuzzification: The final step in an FLS process is output processing. The output processing of type-1 FLSs involves only the defuzzification process. On the other hand, the output processing of IT-2 FLSs involves type reduction followed by defuzzification. Class defuzzify has different operations which perform several kinds of type-reductions and defuzzifications, in accordance with the configuration set by the FLC developers. Class defuzzify is composed of class defuzzify for performing defuzzification.

In the developed library, for all outputs, type-1 Mamdani
Algorithm 2 Inference

Input: configuration, crisp inputs, fuzzified inputs, rules
Output: Output fuzzy set(s)

Begin Procedure

1: Get the fuzzified input;
2: Get configuration;
3: if FLS type is type-1 then
4: Compute the firing level of rules;
5: if Inference is Mamdani then
6: for all outputs do
7: Perform implication on the consequent fuzzy sets;
8: Perform aggregation to get the type-1 output fuzzy set;
9: else # FLS is type-1 TSK
10: for all outputs do
11: Compute the TSK rule outputs and get the output fuzzy set;
12: else # FLS is IT-2
13: Compute the firing level for the upper and lower bounds of the antecedent fuzzy sets of rules;
14: if Inference is Mamdani then
15: for all outputs do
16: Perform implication on the upper and lower bounds of the consequent fuzzy sets;
17: Perform aggregation on the upper and lower bounds of the consequent fuzzy sets to get the output fuzzy set;
18: else # FLS is IT-2 TSK
19: for all outputs do
20: Compute the inner TSK uncertainty bounds ($\mu_l$ and $\mu_r$);
21: Compute the outer TSK uncertainty bounds ($\nu_l$ and $\nu_r$);
22: Compute the upper and lower bounds for the TSK output output fuzzy set ($y_l$ and $y_r$);
23: Return the output fuzzy set;
End Procedure

FLSs are defuzzified using the centroid defuzzification technique while type-1 TSK FLSs are defuzzified using weighted average defuzzification technique.

To obtain a crisp value of IT-2 Mamdani FLSs, for all outputs, the developed library performs Nie-Tan type reduction on the output fuzzy set followed by an IT-2 defuzzification. If the configured FLS is IT-2 TSK FLS, since the developed library uses uncertainty bounds technique, it bypasses the type reduction step and computes the defuzzified crisp output. As shown in Fig. 4, for assisting the defuzzification of user-defined fuzzy sets, class defuzzify is composed of other classes of type-1 and IT-2 MFs. For performing a fast and efficient defuzzification, the developed library performs defuzzification in discrete form.

4) Putting it all Together: The developed FLS library performs fuzzification, inference and defuzzification operations for type-1 and IT-2 FLSs using Algorithm 3. This implementation architecture and algorithm enables the FLC application developers to incorporate multiple FLCs in their applications having independent configurations. The source code of the developed FLS library is available ongithub at https://github.com/ACCESSLab/cpp-fuzzy-library.

Algorithm 3 FLS Library Operations

Input: Crisp Inputs
Output: Crisp Outputs

Begin Procedure

1: Get configuration;
2: while perform_fls = true do
3: Get crisp inputs;
4: Perform fuzzification;
5: Perform inference;
6: Perform defuzzification;
7: Return crisp outputs;
End Procedure

IV. SIMULATING THE UAV ALTITUDE CONTROLLER

To evaluate the effectiveness of the developed library, an IT-2 TSK fuzzy logic altitude control of a quadcopter UAV was simulated in Gazebo STIL simulation environment. For simulating the FLC, the developed library was used.

The IT-2 TSK FLC for the UAV takes two inputs and produces one control output. The inputs are the altitude error, $Z_{err}$, and its derivative, $\dot{Z}_{err}$, where as the control output is the throttle level. The input IT-2 MFs of the TSK FLC are shown in figures 5 and 6. The rule base of the FLC is stated in table I while the output uncertainties of the first-order TSK FLC coefficients ($Z$ and $\dot{Z}$) are stated in table II. For all rules, the TSK coefficient $c_0$ is considered to be zero. The output processing unit used for developing the FLC uses uncertainty bounds aggregation method.

The simulation results are shown in Figures 8 and 9. The results are also compared with a PD controller. To make 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:

\[ y = 0.55 Z_{err} + 0.45 \dot{Z}_{err} \]  

(4)

V. CONCLUSION

This paper has expressed the fuzzy logic system with an object-oriented approach using UML. Additionally, using C++, we developed a generic and configurable FLS library that can ease the implementation of real-time FLCs particularly for robotic applications. Furthermore, a GUI and configuration mechanism was developed to assist FLC application developers for configuring the library. The developed FLC library enables developers to implement multi-input multi-output type-1 and interval type-2 FLCs which are based on Mamdani or TSK inference methodologies. For easing the computation cost of interval type-2 FLSs, the developed library uses Nie-Tan type reduction and uncertainty bounds aggregation methods. Finally, the developed library was used to implement an IT-2 fuzzy logic altitude control of a quadcopter UAV in Gazebo simulation environment.

REFERENCES