IJCSC Volume 5 • Number 1 March-Sep 2014 pp. 202-204 ISSN-0973-7391

FLCPlus: A Light-Weight Open-Source Library In C++ For Fuzzy Logic Controller

Krishna K Panduru
IMaR Research Gateway, Institute of Technology Tralee, Tralee, Kerry, Ireland
kaushal.panduru@gmail.com

Abstract: Recent trends in software development for intelligent control and robotics have invoked a lot of research organisations to work on them. The philosophy of open source and free software and hardware made research and development of new products became faster and much more efficient. One such intelligent control mechanism is Fuzzy Logic. Fuzzy logic is a multi-valued logic which has many areas of control engineering. In this paper we present FLCPlus, a light-weight fuzzy logic controller library which can be easily implemented on systems based on C/C++. Later a simulation experiment is shown in this paper which uses FLCPlus, where a mobile robot with simple range sensors and a differential-drive mechanism. Several comparisons have also made with various libraries on speed of execution, code size and simplicity of API. And the paper is concluded with future work on the open source library.

Keywords: Fuzzy logic, open-source, GitHub.

I. INTRODUCTION

Writing software for intelligent control systems and robotics is difficult. Different types of intelligent control systems use different methods of programming, but most systems use C language in robotics and software engineering. Intelligent control software must have the ability to perceive and control. To overcome these challenges, many software engineers and researchers have written frameworks to manage complex tasks which help to decrease the production time and development process of experiments or applications. Many such software frameworks include ROS, which is an open source robot operating system which has a structured communication layer for process management [1]. OpenTLD, another open source software framework based on C++ and MatLab, it has a robust capability to track, learn and detect features in a high-definition video [2].

In this paper, Section 2 describers the design goals and driving principles involved in design and development of FLCPlus software, section 3 talks about a basic fuzzy logic controller design and some pointers on how it is implemented in C++. Section 4 and 5 talks about the experimental setup arranged to show the functioning of the software library and a brief design procedure to test and show simulations for the software. And section 6 concludes with results and future work.

II. DESIGN GOALS

The main objective of the software was to provide an easy access to intelligent control algorithms and incorporate in any project.

The goals of this FLCPlus can be summarized as:

- Low memory footprint
- Free and open source
- Modular

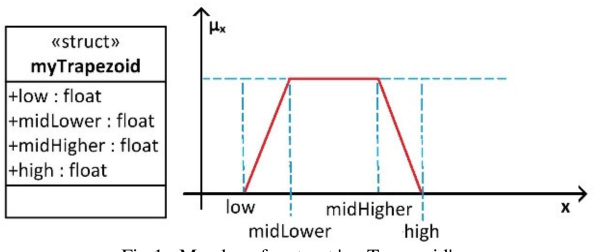


Fig 1: Members for struct 'myTrapezoid'

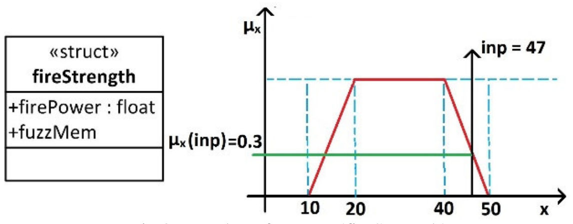


Fig 2: Members for struct 'fireStrength'

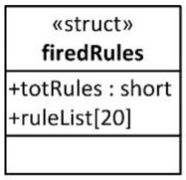


Fig 3: Members for struct 'firedRules'

A. *Low memory footprint*

Major memory issues arise with the use of many input and output membership functions, FLCPlus uses the same output membership functions to generate a list of rules. This reduces the memory usage and increase code execution speed per fuzzy calculation. Declaring the membership variables as constant, the membership values will be stored in the flash memory of a microcontroller which will save RAM space which could be a huge advantage for small memory microcontrollers. Fig 1 through Fig 3 shows the structure of members used in FLCPlus.

B. *Modular*

Since most of the practical applications will not be using more than 4 input, the inference engine can process up-to 4 inputs and unlimited number of membership functions. The number of rules to be written is governed by equation

(1). Generally, context blending of behaviours will be applied to process information with different number of fuzzy sets. The rule base for each fuzzy logic module is implemented using an array of structures. This reduces the need to modify codes on other files.

$$n\ of\ rules = (n\ of\ input)^{n\ of\ variables} \qquad (1)$$

III. FUZZY LOGIC CONTROLLER

Fuzzy logic controller uses fuzzy sets to process information, real world data is converted into fuzzy sets by a process known as fuzzification.

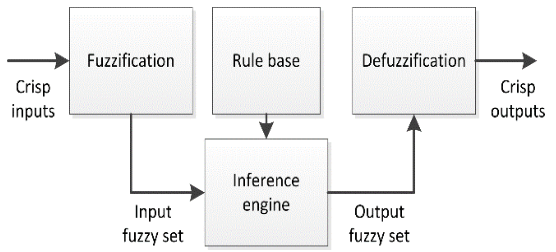


Fig 4: Fuzzy Logic Controller Block Diagram

Fuzzy sets consist of membership functions which are unique to each input. In FLCPlus, the fuzzy membership is defined by a set of floating type data which can form a trapezoid or triangular membership function. Each membership function has a firing power which is obtained by the crisp input value and the membership function.

A list of fuzzy rules is defined in the system which maps the fuzzy inputs to fuzzy outputs. The FLCPlus uses an array of output membership functions which is mapped with respect to various combinations of input membership functions.

$$R_1: if\ (x_0\ is\ A_0\ and\ x_1\ is\ A_1\ \ldots and\ x_n\ is\ A_n)\ THEN\ (z\ is\ B) \qquad (2)$$

The inference engine uses conorms and norms. The logical connective AND is implemented as a minimum operator and OR is implemented as a maximum operator. Various operations can be also used depending on the system and computation requirements.

After obtaining fuzzy output sets which are fired from the inference engine, fuzzy sets are converted into crisp real world values by defuzzification. Many kinds of defuzzification methods exist; there is no one-shot answer for a perfect method. As pointed out in [3] the results obtained from various defuzzification methods are different and selection of a defuzzification method widely varies on the application in hand and computation power available in the system.

IV. EXPERIMENTAL SETUP

In our experimental setup, we have used MobileSim: mobile robot simulator using the ARIA advanced robot interface for application to simulate a differential-drive robot with size frontal sonar sensors. P3DX-SH is a mobile robot modelled after P3-DX robot from adept mobile robots [4]. MobileSim is designed based on Stage simulator with minor modifications by Adept mobilerobots [5]. Custom or predefined map data can be used to simulate 2D worlds. The simulated mobile robot has a differential drive mechanism and 16 range sensors in total, for our experiment we will be using six sonar sensors from the left and front side. The P3DX-SH exhibits left wall following behaviour in our application.

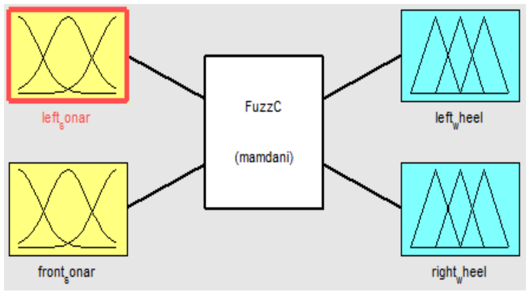
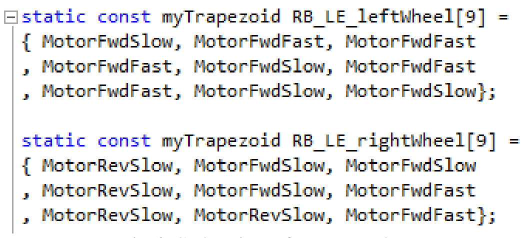


Fig 5: Fuzzy system for simulation

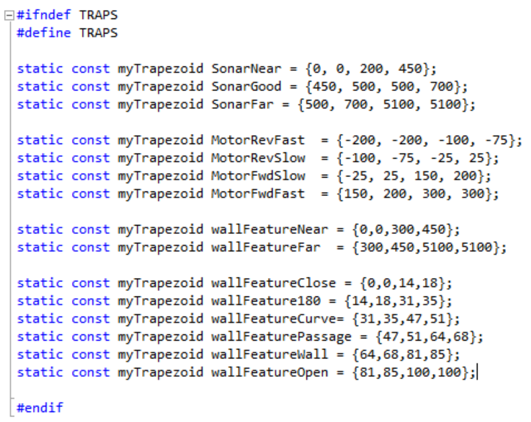
For our fuzzy control system we have used two sets of inputs for left-side sonar sensors and front-side sonar sensors which will be our fuzzy input members and the speed on left and right wheel as out fuzzy output members. Each input fuzzy member has three membership functions, by using equation (1) the number of rules needed for our system will be 9 for each fuzzy output member.

Fig 6 and Fig 7 shows how the membership functions and rule base are written in FLCPlus. In any fuzzy controller membership function and fuzzy rules play a major role in defining how the system works.



```
static const myTrapezoid RB_LE_leftWheel[9] =
{ MotorFwdSlow, MotorFwdFast, MotorFwdFast
, MotorFwdFast, MotorFwdSlow, MotorFwdFast
, MotorFwdFast, MotorFwdSlow, MotorFwdSlow};

static const myTrapezoid RB_LE_rightWheel[9] =
{ MotorRevSlow, MotorFwdSlow, MotorFwdSlow
, MotorRevSlow, MotorFwdSlow, MotorFwdFast
, MotorRevSlow, MotorRevSlow, MotorFwdFast};
```

Fig 6: Code snippet for Fuzzy rules

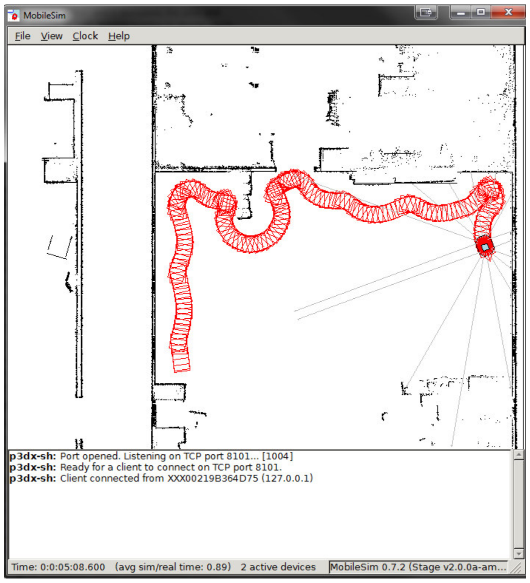


```
#ifndef TRAPS
#define TRAPS

static const myTrapezoid SonarNear = {0, 0, 200, 450};
static const myTrapezoid SonarGood = {450, 500, 500, 700};
static const myTrapezoid SonarFar = {500, 700, 5100, 5100};

static const myTrapezoid MotorRevFast  = {-200, -200, -100, -75};
static const myTrapezoid MotorRevSlow  = {-100, -75, -25, 25};
static const myTrapezoid MotorFwdSlow  = {-25, 25, 150, 200};
static const myTrapezoid MotorFwdFast  = {150, 200, 300, 300};

static const myTrapezoid wallFeatureNear = {0,0,300,450};
static const myTrapezoid wallFeatureFar  = {300,450,5100,5100};

static const myTrapezoid wallFeatureClose = {0,0,14,18};
static const myTrapezoid wallFeature180 = {14,18,31,35};
static const myTrapezoid wallFeatureCurve= {31,35,47,51};
static const myTrapezoid wallFeaturePassage = {47,51,64,68};
static const myTrapezoid wallFeatureWall = {64,68,81,85};
static const myTrapezoid wallFeatureOpen = {81,85,100,100};

#endif
```

Fig 7: Code snippet for fuzzy membership functions

V. RESULTS

Fig 8 shows the trail path obtained by running our software; the trail shows smooth movement of robot in the arena.



```
p3dx-sh: Port opened. Listening on TCP port 8101... [1004]
p3dx-sh: Ready for a client to connect on TCP port 8101.
p3dx-sh: Client connected from XXX00219B364D75 (127.0.0.1)
```

```
Time: 0:0:05:08.600   (avg sim/real time: 0.89)   2 active devices   MobileSim 0.7.2 (Stage v2.0.0a-am...
```

Fig 8: Simulating a P3DX-SH robot in MobileSim

A simulation of fuzzy control surfaces is seen in figure 9 and figure 10. The fuzzy control surfaces shows a smooth transition of output which cannot be obtain so easily by any other controller.

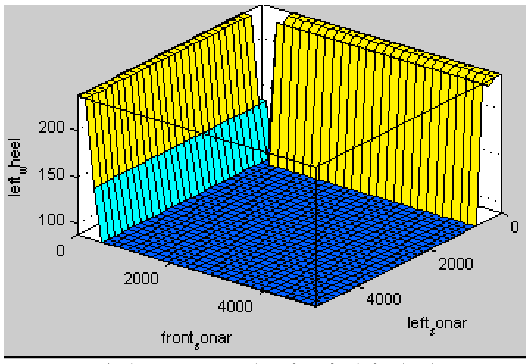


Fig 9: Fuzzy control surface for left wheel

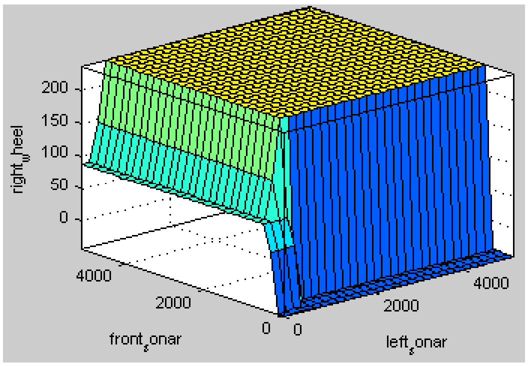


Fig 10: Fuzzy control surface for right wheel

VI. CONCLUSION AND FUTURE WORK

We have designed FLCPlus to support our philosophy of free software. We have also demonstrated a use case example for our library. FLCPlus performed very smoothly on our robot simulation. We anticipate the interesting features such as its open source and free software nature will help other researchers involved in fuzzy logic to take advantage of this library and fork its repositories to apply FLCPlus in their own projects.

The Fuzzy Logic Control library is released under GNU GPL v3 license and is available at GitHub under the project name FLCPlus. For our future work, we will be working on FLCPlus to include more fuzzy membership shapes and defuzzification methods.

VII. REFERENCES

[1] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Ng, “ROS: an open-source Robot Operating System,” *ICRA workshop on open source software,* 2009.

[2] Z. Kalal, K. Mikolajczyk and J. Matas, “FACE-TLD: TRACKING-LEARNING-DETECTION APPLIED TO FACES,” in *International Conference on Image Processing*, 2010.

[3] S. and H. Haidarian, “Towards Autonomous descision making in multi-agent environments using fuzzy logic,” in *CEEMAS*, Prague, 2003.

[4] A. M. robots, “ARIA,” adept mobile robots, [Online]. Available: http://robots.mobilerobots.com/wiki/ARIA. [Accessed 26 March 2014].

[5] L. A. Zadeh, “Fuzzy sets,” *Information and control,* pp. 338-353, 1965.

[6] K. ChangBum, S. Kyoung A., L.-K. Hyung and K. Jeong O., “Design and Implementation of a Fuzzy Elevator Group Control System,” in *IEEE Transcations on Systems, Man, and Cybernetics*, 1998.

[7] B.-J. Choi , R. C. H. Sang-Wan and H. Suk-Kyo, “Refrigerator Temperature Control Using Fuzzy Logic and Neural Network,” in *IEEE International Symposium on Industrial Electronics*, Pretoria, 1998.

[8] A. M. Community, “Adept MobileRobots,” [Online]. Available: http://robots.mobilerobots.com/wiki/MobileSim. [Accessed 26 March 2014].

[9] E. Aguirre and . A. González, “Fuzzy behaviors for mobile robot navigation: design, coordination and fusion,” *International Journal of Approximate Reasoning,* vol. 25, no. 3, pp. 255-289, 2000.

[10] A. Saffiotti, “Fuzzy Logic in Autonomous Robotics: behavior coordination,” in *Sixth IEEE Intl. Conference on Fuzzy Systems*, 1997.