Engineering Applications of Artificial Intelligence 26 (2013) 162–172

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Defuzzification block: New algorithms, and efficient hardware and software implementation issues

H.R. Mahdiani [a,b,*], A. Banaiyan [d], M. Haji Seyed Javadi [c], S.M. Fakhraie [b], C. Lucas [b]

[a] ECE Department, Sh. Abbaspour University of Technology, Tehran, Iran
[b] School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
[c] Department of Computer, Electronics and IT, Qazvin Branch, Azad University, Qazvin, Iran
[d] Department of Computer Science, University of California, Irvine, USA

ARTICLE INFO

Article history:
Received 23 October 2011
Received in revised form
6 May 2012
Accepted 1 July 2012
Available online 20 July 2012

Keywords:
Fuzzy hardware
Defuzzification
VLSI
Fuzzy software
Fuzzy control

ABSTRACT

The defuzzification is a critical block when implementing a fuzzy inference engine due to different variations and also high computational power demands of defuzzification algorithms. These various methods stand for different cost-accuracy trade-off points. Three new implementation friendly defuzzification algorithms are presented in this paper and compared with a complete set of existing defuzzification methods. Some accuracy analysis simulation results and analytic studies are provided to demonstrate that these methods provide acceptable precision with respect to other existing methods. The software models of the proposed and exiting defuzzification methods are developed under three well-known platforms, Intel's Pentium IV, IBM's PowerPC, and TI's C62 DSP to show that new methods gain much lower execution-time and instruction-count with respect to the most common existing methods. The hardware models of all these methods are also developed and synthesized to demonstrate the superiority of the new methods in terms of area, delay, and power consumption with respect to other methods when implemented in hardware.

1. Introduction

Fuzzy control is probably the most important practical fuzzy logic application. Fuzzification and defuzzification processes are the main interfaces of the fuzzy system to the real world, when is used in a real-world application (Sanchez-Solano et al., 2007). The main operation of these two processes is to provide the numeric to linguistic conversion and vice versa (Roychowdhury and Pedrycz, 2001) which make it possible to bind the fuzzy inputs and outputs of the fuzzy inference engine to common sensors and actuators. Although there are many different defuzzification methods, certain types of defuzzification methods can be properly used only for certain types of applications (Runkler, 1997). From the perspective of fuzzy control however, a defuzzification method is acceptable whenever it can be used to solve a practical problem (Roychowdhury and Pedrycz, 2001). The most popular defuzzification technique is the true center of gravity (COG) which is the preferred choice in most applications due to some of its unique features. It provides better accuracy with respect to other defuzzification methods and also is able to track the continuous changes of the system inputs and provide smooth and continuous output values (Broekhoven and Baets, 2004). However, its main disadvantage is its high latency and also computational complexity that prevents to use it in many real world applications (Patel and Mohan, 2002).

A fuzzy inference engine might be implemented in three different approaches: software-only, hardware-only, or any combination of software and hardware (Del Campo et al., 2008; Reyneri, 2003). According to this classification, three different implementation alternatives of a fuzzy system can be identified (Costa et al., 1995): software programs running on general purpose processors or digital signal processors (Baturone et al., 2008; Costa et al., 1995; Leottau and Melgarejo, 2010; Lin and Shen, 2006; Mohagheghi et al., 2009), software programs running on an application-specific instruction processor (Ansari and Gupta, 2009; Banaiyan et al., 2006a, 2006b; Costa et al., 1995), and fully hardware implementation of the system using dedicated fuzzy coprocessors and processors (Basterretxea et al., 2006, 2007; Bulla et al., 2008; Cao et al., 2006, 2009; Hamzeh et al., 2008; Konstantinos et al., 2009; Kwon et al., 2010; Louverdis and Andreadis, 2003; Melgarejo and Pena-Reyes, 2007; Oh et al., 2011). Either to select any of these approaches to implement a fuzzy system based on the application type or performance requirements (Costa et al., 1995; Reyneri, 2003), the designer should select the most appropriate defuzzification method according to the application context and also the trade-off between accuracy (i.e. defuzzified output error) and computational power demand (i.e. hardware complexity or software execution time).

* Correspondence to: Computer and Electronics Group, Sh. Abbaspour University of Technology, East Vafadar Blvd., Tehranpars, P. O. Box: 16765-1719, Tehran, Iran. Tel.: +98 21 73932630; fax: +98 21 77312798.
E-mail addresses: mahdiani@gmail.com, mahdiany@ut.ac.ir, mahdiani@pwut.ac.ir (H.R. Mahdiani).

0952-1976/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2012.07.001

There is some work to guide the designer through choosing the most proper defuzzification method based on the application context (Broekhoven and Baets, 2004) which is beyond the scope of this paper. Also there are some reports which determine the "accuracy-cost" trade-off for different defuzzification methods in software (Broekhoven and Baets, 2004; Lancaster and Wierman, 2003) or hardware (Phongpensri and Sripanomwan, 2009). Knowing this trade-off guides the designer to choose the best suitable defuzzification method which best suites the precision requirements with minimum computational power demand which results in better area, delay and power consumption in hardware or less execution time in software implementations.

In this paper, three new defuzzification methods are introduced which preserve the same accuracy of the existing methods while providing much better implementation gains. An all-inclusive survey of some well-known existing defuzzification methods, as well as the description of the new methods is presented in the following section. Section 3 presents the simulation results and theoretical accuracy analysis studies to compare the accuracy of the proposed and existing defuzzification methods. The software and hardware implementation aspects of new and traditional defuzzification methods are discussed in Sections 4 and 5. The last section concludes the paper.

2. A survey of existing and new defuzzification methods

All existing defuzzification methods might be categorized into standard or non-standard categories (Roychowdhury and Pedrycz, 2001). The standard methods are mostly preferred due to their feasible implementations while on the other hand, the non-standard defuzzification methods such as BADD (Yager and Filev, 1991; Yager and Filev, 1993a) or SLIDE (Yager and Filev, 1993b) are theoretically developed and are not widely used in fuzzy applications (Roychowdhury and Pedrycz, 2001). As we deal with the implementation aspects of the defuzzification methods, the non-standard methods are not included in this paper. Also as there is no unique convention about the naming of different standard defuzzification methods, some of them are referenced with different names by different authors. To address this problem and clarify the paper terminology, a brief description of the most important standard defuzzification methods is presented in the following and the three new defuzzification methods are introduced at the end of this section.

2.1. Existing standard defuzzification methods

The standard defuzzification algorithms which are mostly used in experimental and industrial fuzzy control applications, fundamentally focus on geometric area-based computations (Roychowdhury and Pedrycz, 2001). These methods provide different accuracy-computational power trade-offs and might be easily implemented in hardware or software. This is the main reason for their popularity. Some of these methods such as mean of maxima are ad hoc and are not supported by proper analytical reasoning and so, are suitable only for some specific applications. Some other standard methods such as center of gravity and center of sum however; should be considered as classic methods with convenient scientific reasoning.

2.1.1. True center of gravity (COG) (Broekhoven and Baets, 2004; Saade and Diab, 2000)

The center of gravity which is also named as the Center of Area (COA) in some references (Roychowdhury and Pedrycz, 2001; Tao and Yao, 1996; Watanabe et al., 1991) is a simple and elegant method. To compute the COG output, all the rules should be processed first and all the fuzzy membership functions (MFs) on THEN sides of the fired rules should be combined together to form the unified output fuzzy shape (Fig. 1a). The final defuzzified output value is the center of gravity of this unified shape which can be computed by exploiting discretization method. In this method, the whole area is divided into narrow rectangles around $(x_i, \mu(x_i))$ center point, to simplify the center of gravity computation process. The Eq. (1) demonstrates the COG computation using discretization method

$$\mathrm{COG} = \frac{\sum_{i=1}^{k} x_i \mu(x_i)}{\sum_{i=1}^{k} \mu(x_i)} \tag{1}$$

where 'k' is the number of discretization levels. Higher discretization levels imply narrower rectangles which the unified shape is divided into, and so the closer value to the real center of gravity (Broekhoven and Baets, 2004). This method is very accurate (Lancaster and Wierman, 2003) and is most easily implemented in software (Lees et al., 1996). As well, there are some hardware realizations of the COG method (Lees et al., 1996; Louverdis and Andreadis, 2003; Reyneri, 2003; Watanabe et al., 1990, 1991). However it has some important disadvantages. The most important

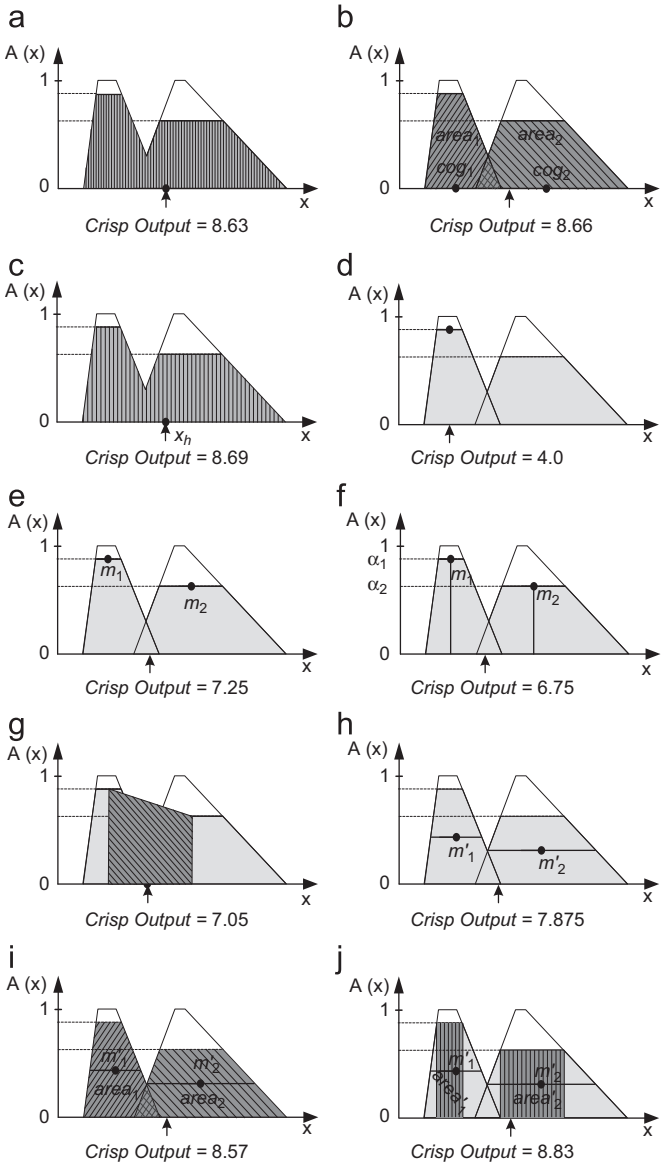

Fig. 1. Illustration of different defuzzification methods: (a) COG, (b) COS, (c) MOA, (d) MOM, (e) PA, (f) WPA, (g) Sparus, (h) TMA, (i) TWTMA, and (j) RWTMA.