ATLANTIS PRESS International Journal of Computational Intelligence Systems, Vol. 10 (2017) 13–22

FISDeT: Fuzzy Inference System Development Tool

Giovanna Castellano, Ciro Castiello, Vincenzo Pasquadibisceglie, Gianluca Zaza

Computer Science Department, University of Bari "A. Moro",
Via Orabona, 4,
Bari, 70125, Italy
E-mail: {giovanna.castellano,ciro.castiello}@uniba.it,
{vincenzo.pasquadibisceglie,gianlucazaza92}@gmail.com

Received 3 March 2016

Accepted 8 August 2016

Abstract

This paper introduces FISDeT, a tool to support the design of Fuzzy Inference Systems, composed of a set of Python modules sharing the standard specification language FCL used for FIS definition. FISDeT includes a graphical user interface that enables easy definition and quick update of elements composing the knowledge base of a FIS. Given the knowledge base, the tool can perform the inference of fuzzy rules, providing the output of a FIS for any given input. Modules for creating a fuzzy rule base for classification and verifying the behavior of the classification system are integrated within the environment. The paper also reports applications of FISDeT to acquire available knowledge bases and create a FIS for an image processing task.

Keywords: Fuzzy Inference System, Python, FCL, Open Source Software.

1. Introduction

A Fuzzy Inference System (FIS) is an extension of a classical rule-based system, having IF-THEN rules with antecedents and consequents composed by fuzzy logic statements[1,2]. A FIS is made up of a Knowledge Base, that comprises the information used by the human expert in the form of linguistic rules, and an Inference Engine, that uses the input values and the information from the Knowledge Base to perform a reasoning process and to infer output values. FISs are suitable for solving real-world problems for which it is too difficult to obtain a mathematical model, while it is quite easy to describe the solution linguistically by means of IF-THEN rules. They have been successfully applied in many fields such as decision-making, control[3], image recognition[4] and non-linear system modeling[5].

Despite the power of FISs for solving a wide range of problems, their implementation requires some effort and programming expertise. In the last years, many tools have been developed to enable easy design of a FIS by reducing specific expertise. Some tools have become commercially available, such as the MATLAB Fuzzy logic toolbox (`www.mathworks.com`), others are currently available as open source software, mainly based on Java or C++ languages (see section 2). To the best of our knowledge, no open source tool based on the Python language has been developed so far.

In this paper we present FISDeT (Fuzzy Inference System Development Tool), a novel tool aimed to facilitate the creation of a fuzzy rule-based system. The current release of FISDeT for different platforms is available at `https://github.com/Fisdet/FISDeT`. FISDeT is conceived as a package

implemented in Python on the basis of the *pyfuzzy* library[6]. It includes two main modules, one devoted to the creation of the knowledge base and the other designed to accomplish the inference of fuzzy rules. Key aspects of the proposed tool are userfriendliness and interactivity which allow the user to define and quickly modify the fuzzy knowledge base by means of a simple and intuitive graphical interface. After fuzzy rules have been created and tested, the tool stores the created knowledge base as a source code which follows the standard FCL (Fuzzy Control Language) introduced by IEC 61131[19], thus enabling portability of the created FIS on different inference systems supporting FCL.

The paper is organized as follows. Section 2 provides an overview of other existing tools for FIS design. The FISDeT tool is described in section 3. In section 4 we present the *pyfuzzy* library and the standard language FCL which it supports. We discuss application examples in section 5. Final remarks are presented in section 6.

2. Related works

A number of software tools have been proposed in literature to create and execute a FIS. In this section we highlight the major features of non-commercial tools in order to present FISDeT as a novel alternative in this scenario. We focus our interest on free software distributions since open source software plays an important role in the scientific research community[7]. We start from the analysis reported in Ref. 8 (where twenty-five non commercial packages for FIS design are considered) to outline the main features of existing tools.

Some of existing packages and libraries, such as DotFuzzy[9], FRBS[10], JFuzzinator[11], libFuzzyEngine[12], nxtfuzzylogic[13] enable the design of a FIS for specific purposes. Such specific tools, even if they are simple and easy to use, usually have limited functionality. For example, they include only one membership function (typically trapezoid) and/or one defuzzification method. Hence, general-purpose tools are preferable. FisPRO[14] is a general-purpose tool developed in Java that provides an interactive environment for designing and optimizing fuzzy inference systems. To our knowledge, no tool exists in the Python community that enables the design of general-purpose FISs. In this sense, we propose FISDeT as a novel Python tool for the design of a general-purpose FIS.

Concerning standardization, only few packages support the FCL language defined by the IEC 61131-7 specification. FCL is a structured language oriented to the definition of fuzzy logic-based control systems. Only a limited programming experience is required to learn and understand such a language. A number of packages can be mentioned among those currently supporting FCL: FFLL[15], AwiFuzz[16] and JFuzzyQT[17] (they are all written in C++). However, as stated in Ref. 8, in many cases such tools present problems when trying to read FCL files. With regard to Java tools, JFuzzyLogic[8] is one that correctly supports FCL. Currently there are no tools written in Python that support FCL. To this aim, we propose FISDeT as a novel Python tool supporting the FCL standard.

In short, the main deficiencies of the existing tools are: lack of standardization, difficulty in the use, special-purpose design. These are the limitations that encouraged the development of the FISDeT tool described in this paper. FISDeT has been developed by following the line of the most recent tools, being designed to provide standardization, platform independence and easy use. In addition, it represents the first tentative in the Python developers community to provide a tool supporting standard FIS design.

Recently, in Ref. 18 a deep analysis of existing software for fuzzy system development has been presented. The authors emphasize the four aspects that should guide further research directions in fuzzy systems software in the future, namely *interoperability*, *novelty*, *usability*, and *relevance*. We followed these guidelines to analyze the main strength points of FISDeT. Regarding *interoperability*, it is widely admitted by the fuzzy community that there is a need for a universal standard language to foster the interoperability of software for fuzzy system development[18]. Since FISDeT uses the FCL language to define fuzzy systems, it allows to exchange fuzzy systems across different platforms and

Table 1: Comparison of a number of fuzzy logic tools: the features are related to the presentation proposed in Section 2: programming language for implementation; independence from specific platforms; presence of GUI; support for the FCL standard; general-purpose conception; open source availability.

	Language	Platform independence	GUI	FCL	General-purpose	Open source
AwiFuzz	C++	✓	-	✓	-	✓
DotFuzzy	C#	✓	-	-	-	✓
FFLL	C++	✓	-	✓	-	✓
Fispro	C++/Java	✓	✓	-	✓	✓
FRBS	C++	✓	-	-	-	✓
JFuzzinator	Java	✓	-	-	-	✓
jFuzzyLogic	Java	✓	-	✓	✓	✓
jFuzzyQt	C++	✓	-	✓	✓	✓
libFuzzyEngine	C++	✓	-	-	-	✓
nxtfuzzylogic	Java	✓	-	-	-	✓
FISDeT	Python	✓	✓	✓	✓	✓

thus improves interoperability. The *novelty* of a newly developed tool requires a previous analysis based on the functionality criteria of the existing software in the literature to enable researchers to identify the software needs that the current software cannot provide[18]. We have carried out a preliminary analysis of the existing software in the literature according to basic characteristics such as programming language, interoperability, GUI, use of FCL, general/special purpose, open-source. Table 1 shows a comparison of the FISDeT facilities with other tools reported in literature. From the analysis it emerged that: i) no software for fuzzy systems has been developed using the Python language so far; ii) there are only a few software offering a GUI for easy user interaction. FISDeT is going to introduce in the literature panorama some elements of novelty in both those respects. Regarding *usability*, Ref. 18 states that *"the open-source model makes it easier for other researchers to develop and adapt available software to their problems and provides many advantages."* Moreover, usability is obviously enhanced whenever a tool includes visual support among its facilities. FISDeT is an open-source tool and it is equipped with a properly designed GUI, hence its usability is high. Finally, regarding *relevance*, we made the FISDeT software available on the Github platform so that users can download our software that we are going to keep updated. This kinds of commitments will contribute to the relevance of our tool.

3. Architecture of FISDeT

FISDeT is conceived as a development environment with a GUI that enables easy construction of a FIS and efficient inference of fuzzy rules. Figure 1 shows the general architecture of FISDeT. It is composed of two main modules:

- The **KBC (Knowledge Base Creator)** contains a set of procedures written in Python designed to enable all the steps related to the definition of the fuzzy knowledge base (i.e., the definition of input and output variables, linguistic terms and rules). The KBC stores all these elements in FCL format, which is the common format used inside the environment when handling the FIS components. Besides manual definition of the knowledge base, the user is allowed to import an existing knowledge base in FCL format and eventually to modify its elements.
- The **IE (Inference Engine)** contains a set of procedures included in the *pyfuzzy* library related to

the parsing of a FCL file, the importing of input data and the inference of fuzzy rules. Using a graphical interface, the outcome values of the FIS are graphically shown.

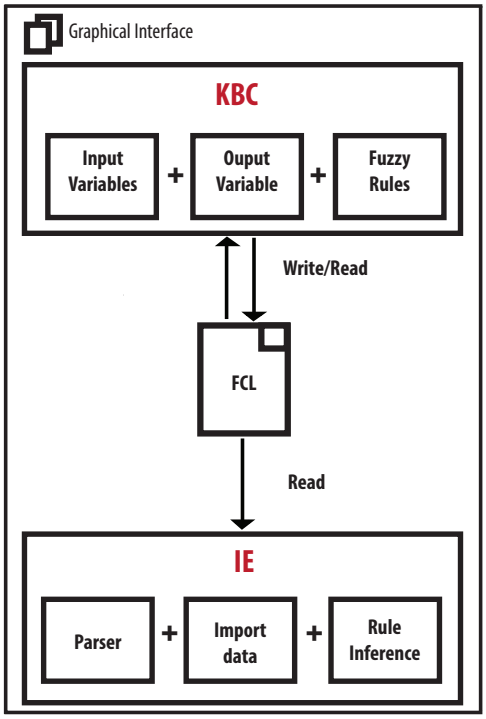

Fig. 1. Architecture of FISDeT.

Both these modules communicate with a GUI providing a simple and intuitive access to the elements of the FIS. A detailed description of the KBC and IE modules is given in the following.

3.1. *The Knowledge Base Creator*

The KBC module is designed as a graphical tool to guide the user through the definition of:

- the input/output variables
- the fuzzy sets associated to each variable
- the fuzzy rules

To this aim, the KBC module includes two areas: the fuzzy variable area and the fuzzy rule area.

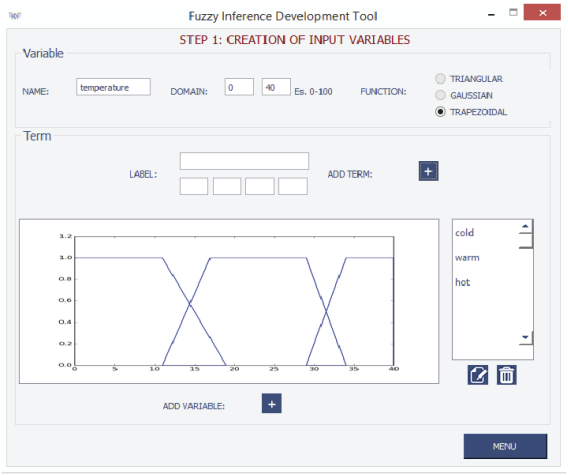

Fig. 2. GUI of the fuzzy variable area for the creation of an input variable.

In the fuzzy variable area the user can define the name of the input/output variables, together with the linguistic terms (fuzzy sets) associated to each variable and the corresponding membership functions. To define fuzzy sets, the user can choose among three predefined types of membership functions (triangular, trapezoidal and Gaussian). With regard to the output variable, a singleton fuzzy set can also be defined. The graphical interface (depicted in Figure 2) presents the plot of fuzzy sets for each variable. The user can define/adjust the values of the membership function parameters by typing new values in the provided text areas, and confirming by clicking on the "ADD TERM" button. A form on the right side lists all terms that are currently assigned to each variable. The user can modify (respectively, delete) a term by selecting it in the list and then by clicking the iconic "EDIT" button (respectively, "TRASH" button). The user is going to repeat this process until all the terms have been assigned to each variable. Once the user has assigned all the terms to a variable, a new variable can be added to the knowledge base by clicking the "ADD VARIABLE" button. While defining the output variable (as reported in Figure 3), the user can choose between different defuzzification methods: Center of Gravity, LM (Local Left Maximum) and RM (Local Right Maximum). They correspond to the methods supported by the *pyfuzzy* library used to implement FISDeT.

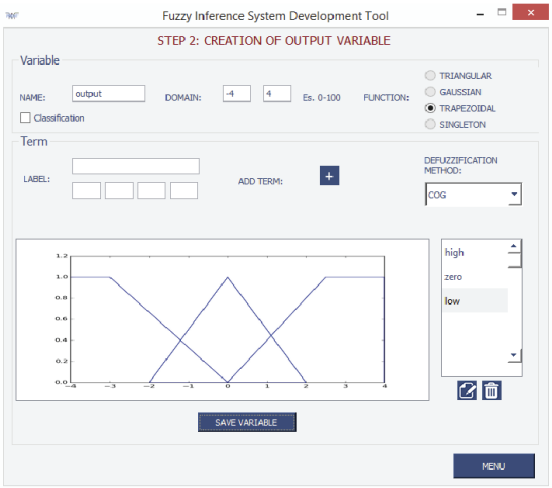

Fig. 3. GUI of the fuzzy variable area for the creation of the output variable.

In the fuzzy rule area (Figure 4) the user can easily define rule's antecedents and consequents: the IF-THEN structure of a fuzzy rule is highlighted on the screen and each antecedent (or consequent) can be defined by choosing a fuzzy term from a combo box. Once defined, a rule can be added to the knowledge base by simply clicking on the iconic "ADD" button. Moreover, the user can delete a rule by using the "DELETE RULE" button. Currently no choice is available for the rule aggregation operator since only one t-conorm operator is implemented, namely the maximum function. With respect to the t-norm operators, the tool implements the minimum function. Finally, the created fuzzy knowledge base can be saved in FCL format.

Besides manual definition of the knowledge base, the KBC module enables the importing of an existing knowledge base stored in FCL format, thus allowing the visualization of its elements. Once the knowledge base is imported, the user can edit or delete any fuzzy set, by simply clicking on it and selecting the proper iconic button. Also, the user can access the list of rules and eventually delete some rules.

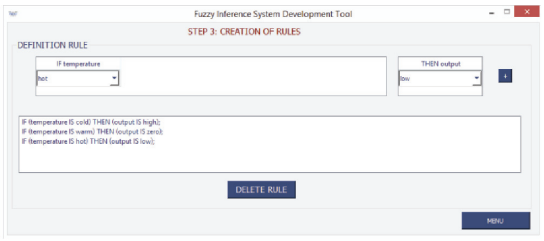

Fig. 4. GUI of the fuzzy rule area.

3.2. *The Inference Engine*

The IE module performs the standard process of rule inference. Such an inference process is accomplished on the basis of a number of subsequent steps. Firstly, the knowledge base is retrieved: the IE captures the FIS configuration by reading the involved parameters stored in a FCL file. Then the user is supposed to specify a set of input values. Finally, the engine applies the process of rule inference and calculates the output value that is brought to the attention of the user. Figure 5 shows the GUI of the inference engine.

The inference process can be also applied to a collection of input data, that can be uploaded by means of the "textscimport data" button. This enables the application of a FIS to a wide range of real-world problems for which observational data are available. Suppose that a FIS has been defined to solve a diagnosis problem. In such a case, it could be useful to perform inference of rules and obtain the outcome of the FIS for a collection of data describing a set of patients. This issue is shown in the experimental section 5.

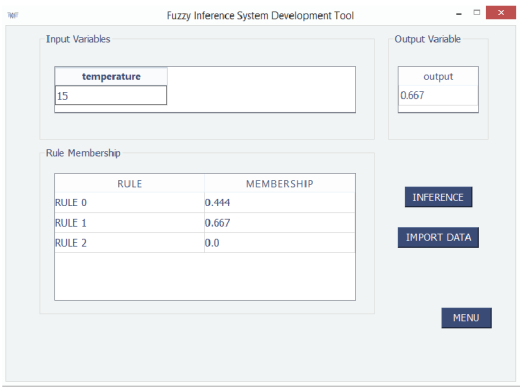

Fig. 5. GUI of the inference engine.

4. Implementation of FISDeT

The implementation of FISDeT is based on the Python language. As concerns the KBN module, we developed a number of classes to implement the procedures pertaining to the fuzzy variable area and the fuzzy rule area. Figure 6 shows the class diagram related to the fuzzy variable area.

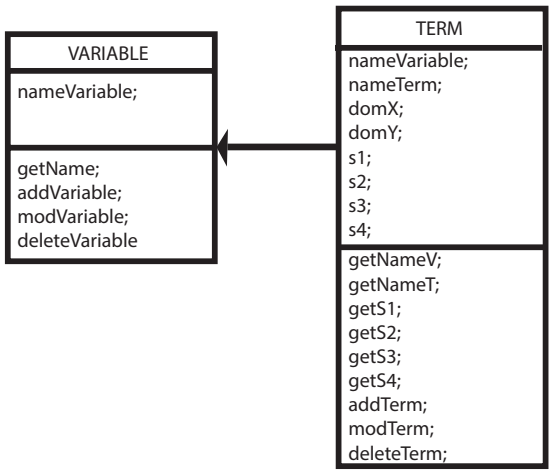

Fig. 6. Class diagram for the implementation of the fuzzy variable area.

The implementation of the IE module is mainly based on *pyfuzzy*[6], an open source library written in Python that allows the design of fuzzy controllers. One main advantage of using this library is the independence of the platform: being written in Python, *pyfuzzy* is compatible with any operating system. Moreover *pyfuzzy* provides a standardization of FIS design to the developers community since it is based on the standard Fuzzy Control Language (FCL) published by the International Electrotechnical Commission (IEC 61131-7)[19]. The IEC-61131 norm is well known for defining the Programmable Controller Languages, commonly used in industrial applications. All IEC-61131 languages are modular. The basic module is called Programmable Organization Unit (POU) and includes Programs, Functions or Function Blocks. A system is usually composed of many POUs, and each of these POUs can be programmed in a different language. A Function Block resembles a very primitive object. It can have multiple input and multiple output variables, can be enabled by an external signal, and can have local variables. Unlike an object, a function block only has one execution block (i.e. there are no methods). The underlying idea for these limitations is that one should be able to implement programs using either text-based or graphic-based languages. Having only one execution block allows to easily control execution when using graphic-based language to interconnect POUs.

FCL uses a single POU type called Fuzzy Inference System (FIS) which is composed of one or more Function Blocks (FBs). Each FB has variables (input, output or instance variables) as well as one or more Rule Blocks (RBs). Each rule block is composed of a set of rules, as well as Aggregation, Activation and Accumulation methods. Aggregation methods play the role of AND, OR and NOT operators for fuzzy sets. These can be Minimum, Product or Bounded difference operators. Activation methods define how rule antecedents modify rule consequents, i.e. once the IF part has been evaluated, how this result is applied to the THEN part of the rule. The most common activation operators are Minimum and Product. Finally, accumulation methods defines how the consequents from multiple rules are combined within a Rule Block. Accumulation methods defined in the standard include: Maximum, Bounded sum, Normed sum, Probabilistic OR, and Sum. All methods defined in the standard are implemented in *pyfuzzy*. The code reported in the following lines shows a simple FIS described using FCL.

```
FUNCTION_BLOCK dummy
    VAR_INPUT
        temperature : REAL; (* RANGE(0 .. 40) *)
    END_VAR

    VAR_OUTPUT
        output : REAL; (* RANGE(-4 .. 4) *)
    END_VAR

    FUZZYFY temperature
        TERM cold := (0, 0) (0, 1) (10, 1) (19, 0) ;
        TERM warm := (11, 0) (17, 1) (29, 1) (34, 0) ;
        TERM hot := (29, 0) (34, 1) (40, 1) (40, 0) ;
    END_FUZZYFY

    DEFUZZYFY output
        TERM high := (-4, 0) (-4, 1) (-3, 1) (0, 0) ;
        TERM zero := (-2, 0) (0, 1) (0, 1) (2, 0) ;
        TERM low := (0, 0) (2.5, 1) (4, 1) (4, 0) ;
        ACCU:MAX;
```

```
        METHOD:MaxRight;
        DEFAULT := 0;
    END_DEFUZZYFY

    RULEBLOCK temperature
        AND:MIN
        RULE 0:IF (temperature IS cold) THEN (output IS
high);
        RULE 1:IF (temperature IS warm) THEN (output IS
zero);
        RULE 2:IF (temperature IS hot) THEN (output IS
low);
    END_RULEBLOCK
END_FUNCTION_BLOCK
```

Only two membership functions are defined in the IEC standard: singleton and piece-wise linear. The *pyfuzzy* library also implements other commonly used membership functions: trapezoidal, sigmoidal, and Gaussian. *Pyfuzzy* implements the standard fuzzy inference process according to the Mamdani-type model. Besides different rule aggregation and activation methods, some defuzzification methods are supported, which are 'Center of gravity', 'Rightmost Max', 'Center of area', 'Leftmost Max', 'Mean max' (for continuous membership functions), or 'Center of gravity' (for discrete membership functions).

To perform the parsing of FCL code we used ANTLR v3[20]. This is a powerful parser generator for reading, processing, executing, or translating structured text of grammatical descriptions containing actions in a variety of target languages, including FCL.

As described in this section, FISDeT is grounded on the *pyfuzzy* library. It should be pointed out, however, that the implementation of a GUI allows the FISDeT user to disengage from code as well as to avoid the employment of a Python IDE where some specific library must be imported. The adopted GUI enables a guided approach for building up an inference system without writing lines of code or descending into the details of some particular library.

5. Application examples

In order to illustrate the usage of FISDeT we consider a couple of employment examples. The first one concerns the use of FISDeT for the acquisition and the modification of a FIS whose knowledge base was previously extracted in an automatic way from a set of data representative of a classification problem. The second one is an example of FIS definition, i.e. the creation of a knowledge base "from scratch" by exploiting only the human experts' knowledge.

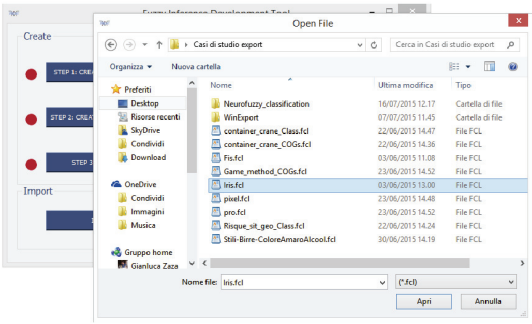

Fig. 7. Importing FCL code in FISDeT.

5.1. *Importing a FIS*

The first example aims at showing the capability of the FISDeT tool to visualize and analyze the knowledge base of a given FIS. In this example we employed the well-known Iris dataset which deals with classification of Iris flowers and a previously defined FIS that was automatically generated from data by using the DC* algorithm[21]. DC* automatically extracts interpretable fuzzy rules from data by means of a double clustering process. This process preserves the multidimensional relations among data and allows to define both the minimal required granulation level for each input feature without further tuning mechanisms. The only required parameter for DC* is the number of prototypes needed for the first clustering phase, corresponding to the maximum number of fuzzy rules in the rule base. Obviously DC* is adopted here for the sake of example: any other algorithm for rule extraction can be used as well, provided that a rule representation can be obtained in the FCL standard.

Once translated in the FCL format,* the extracted fuzzy rules were imported in FISDeT (Figure 7). After the importing operation, all the FIS components are available for visualization and the inference process can be performed. FISDeT can provide the inference result for a single input sample as well as for a collection of input samples, that can be imported using the "IMPORT DATA" facility. This is very useful when a set of observational data is available and the user wants to obtain all the outcomes in a single step (e.g. medical diagnosis of a group of patients). If a manual tuning is required to improve a FIS automatically acquired from data, FISDeT allows to operate both on the rule base and on the membership functions (with the possibility to add/remove a rule or a fuzzy set and to modify parameters/shapes of membership functions).

5.2. *Creating a FIS*

The second example concerns a common problem of image processing, namely contrast enhancement, which is one of the principal intensity transformations that can be applied to a digital image.

According to Ref. 22, we can express the task of enhancing the contrast of a gray-scale image by means of the following fuzzy rules:

> IF a pixel is dark THEN make it darker
> IF a pixel is gray THEN make it gray
> IF a pixel is bright THEN make it brighter

where the input variable *pixel* represents the intensity value of a pixel defined on the interval $[0, 255]$, and the concepts *dark*, *gray* and *bright* are represented by fuzzy sets defined on this domain. Figure 8 shows the use of FISDeT to define the membership functions of these fuzzy sets. As concerns the output variable, it represents the adjustment of intensity expressed as a percentage. Hence the concept *darker* indicates a percentage of a dark intensity value (100% being the limiting shade of dark, i.e. full dark), the concept *brighter* represents a percentage of a bright shade (100% being the limiting value, i.e. full white) and the concept *gray* represents degrees of an intensity in the middle of the gray scale. When interpreted as constant intensities whose strength is modified, the output membership functions are singletons. Figure 9 shows the use of FISDeT to define singletons on the output variable.

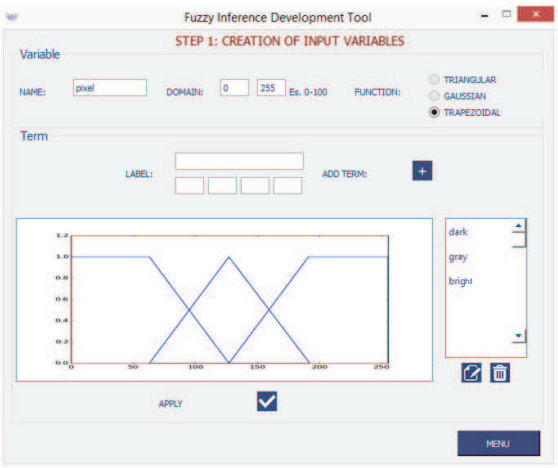

Fig. 8. Definition of fuzzy sets for the input variable *pixel* using FISDeT.

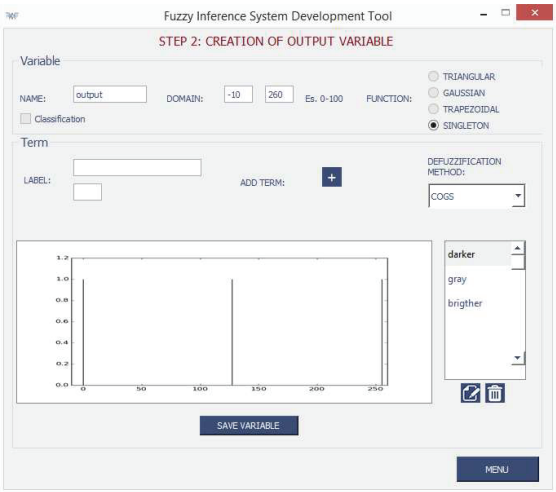

Fig. 9. Definition of fuzzy singletons for the output variable using FISDeT.

Finally, we defined fuzzy rules using the interface provided by FISDeT (Figure 10). Once completed the definition of the knowledge base, we

* The knowledge base extracted by DC* was translated in the FCL standard by an additional Python module purposely designed.

saved it as an FCL file.† The resulting FIS can be used to enhance the contrast of any gray-level image. Given the pixel intensities of an image, FISDeT performs inference of rules and provides new intensity values for the enhanced image. Figure 11 shows the result of applying the inference of the defined FIS to a couple of images.

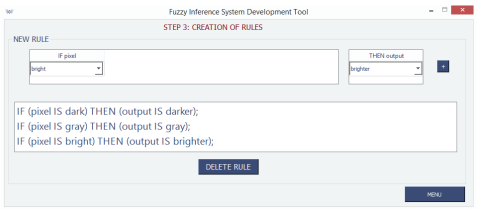

Fig. 10. Definition of fuzzy rules for the contrast enhancement using FISDeT.

Fig. 11. (a) Low-contrast image. (b) Enhanced image obtained by inference of the FIS defined for contrast enhancement.

6. Conclusions

In this paper, we described FISDeT, a software tool for the design of a FIS following the standard IEC 61131. FISDeT brings the benefits of open source software and standardization to the fuzzy systems community. As application examples, we used FISDeT both to acquire available fuzzy knowledge bases (related to a specific classification problem) and to design a FIS purposely conceived to tackle a specific task (related to the problem of image contrast enhancement), thus showing how FISDeT can be used to easily implement a FIS from scratch. Experimentation with the tool on such application examples leads to the conclusion that FISDeT offers a very simple and straightforward way to develop fuzzy rule-based systems for a broad variety of problems.

The effectiveness of the tool lays on an accurate design which focuses on user-friendliness and interactivity. Summarizing, its main advantages are:

- Standardization. FISDeT is based on the *pyfuzzy* library which contains the basic programming elements for the standard IEC 61131-7, which reduces programming work.
- Extensibility. FISDeT is an open source environment, which is of special interest for the research community.
- Platform independence. FISDeT can be run on any hardware and operating system configuration that supports Python.
- User-friendliness. The graphical interface reduces programming work and extends the range of possible users applying fuzzy systems.

The FISDeT software package is continuously being updated and improved. The current release of FISDeT for different platforms is available at `https://github.com/Fisdet/FISDeT`. By now the tool enables the construction of MISO Mamdani-type fuzzy systems. Further work is in progress to extend FISDeT so as to enable the definition of MIMO fuzzy systems.

† The FCL code for this case study is available for download at `https://github.com/Fisdet/FISDeT`.

References

1. L. A. Zadeh. The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets Syst.* **11**(1–3) (1983) 197–198.
2. R. Yager and D. Filev, *Essentials of fuzzy modeling and control*, (Wiley, NY, 1994).
3. P. Bonissone, Fuzzy logic controllers: An industrial reality, in *Computational Intelligence: Imitating Life*, (IEEE Press, Piscataway, NJ, 1994), pp. 316–327.
4. J. C. Bezdek, J. Keller, R. Krisnapuram and N. R. Pal, *Fuzzy models and algorithms for pattern recognition and image processing*, (Springer Science and Business Media, NY, 2005).
5. O. Demir, I. Keskin and S. Cetin, Modeling and control of a nonlinear half-vehicle suspension system: A hybrid fuzzy logic approach. *Nonlinear Dyn.* **67**(3) (2012) 2139–2151.
6. "Pyfuzzy API Documentation" 19 Oct 2009.
7. S. Sonnenburg, M. Braun, C. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, K. R. Muller, F. Pereira, C. Rasmussen, G. Ratsch, B. Scholkopf, A. Smola, P. Vincent, J. Weston and R. Williamson. The need for open source software in machine learning. *J. Mach. Learn. Res.* **8** (2007) 2443–2466.
8. P. Cingolani and J. Alcalá-Fdez. jFuzzyLogic: a java library to design fuzzy logic controllers according to the standard for fuzzy control programming. *Int. J. Comput. Int. Sys.* **6** (2013) 61–75.
9. DotFuzzy. `https://github.com/MicheleBertoli/DotFuzzy` [28 July 2015]
10. FRBS - Fuzzy rule-based systems. `http://dicits.ugr.es/software/FRBS/` [28 July 2015]
11. JFuzzinator. `http://sourceforge.net/projects/jfuzzinator/` [28 July 2015]
12. LibFuzzyEngine++. `http://sourceforge.net/projects/libfuzzyengine/` [27 July 2015]
13. nxtfuzzylogic. `www.openhub.net/p/nxtfuzzylogic` [27 July 2015]
14. S. Guillaume and B. Charnomordic. Fuzzy inference systems: An integrated modeling environment for collaboration between expert knowledge and data using FisPro. *Expert Syst. Appl.* **39**(10) (2012) 8744–8755.
15. FFLL. Free Fuzzy Logic Library. `http://ffll.sourceforge.net/` [27 July 2015]
16. AwiFuzz. Fuzzy Logic Control System. `http://sourceforge.net/projects/awifuzz` [27 July 2015]
17. jFuzzyQt. C++ Fuzzy Logic Library. `http://sourceforge.net/projects/jfuzzyqt/` [27 July 2015]
18. J. Alcala-Fdez, J.M. Alonso. A Survey of Fuzzy Systems Software: Taxonomy, Current Research Trends and Prospects. *IEEE Transactions on Fuzzy Systems.* **24**:1 (2016) 40–56.
19. International Electrotechnical Commission technical committee industrial process measurement and control. IEC 61131 - Programmable Controllers - Part 7: Fuzzy control programming. IEC, 2000.
20. T. Parr. *The definitive ANTLR reference: building domain-specific languages*, (Pragmatic Bookshelf, 2007).
21. M. Lucarelli, C. Castiello, A.M. Fanelli and C. Mencar, Interpretable knowledge discovery from data with DC*, in *Proc. 2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15)*, (Atlantis Press, 2015), pp. 815–822.
22. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, (Prentice Hall, Upper Saddle River, NJ, 2002).