DESIGNING A LIBRARY FOR COMPUTING [PERFORMANCES] WITH WORDS

Alain BONARDI

IRCAM-CNRS, UMR 9912 – STMS
1, place Igor-Stravinsky, 75004 Paris, France
E-mail: alain.bonardi@ircam.fr

Isis TRUCK

LIASD – EA 4383, University Paris 8
2 rue de la Liberté, Saint-Denis Cedex, 93526, France
E-mail: truck@ai.univ-paris8.fr

In this paper we focus on the interaction between performers and machines using linguistic tools. Indeed the Computing with Words (CW) framework is well-suited to human performances when linguistically expressed. We thus address the problem of designing a library that implements such concepts in the performing arts framework. The proposal takes place in the Max/MSP software where our *patches* have been included. A concrete example is given about how to use them to qualify performers observations in the theatre play *Les petites absences*.

Keywords: performing arts, computing with words, Max/MSP *patches*, fuzzy library for linguistic processing.

1. Introduction

Performing arts on stage have been for centuries a place of technical innovation.[8] For the last twenty years, computers have been used for various and numerous purposes: lightning and set command, image and sound generation and projection, etc.

This automation movement led to nearly fully digital devices for stage, but it nearly did not take into account performer to machine dialogue, especially all developments in the field of realtime interaction computing. In the middle of the 1980's the first environment for music realtime processing appeared, named *Patcher*. The purpose was to create interesting and lively musical interaction between human musicians and electronic sets of sound

transformations so that the rigid model of tape accompaniment could be overcome. It gave birth a few years later to the reference software *Max* (that became Max/MSP/Jitter) and later to *PureData*. Since then many people have used them and a real community has appeared with a lot of contributors to Max/MSP; the paradigm has spread to all performing arts: dance, theatre, opera, etc.

In this paper we are interested in managing linguistic data into Max/MSP to be able to "process" human performances. But implementing such high-level concepts into a rather low-level environment is not obvious: this is the point addressed in Section 3. In Section 4 we describe a use-case of our *FuzzyLib* library for Max/MSP. We then conclude by pointing out the main contributions and the future works. All scenic examples come from the theatre play *Les petites absences*, created in December 2008 in *La Comédie de Caen* by *Le Théâtre du Signe*.

2. Background and Related Works

The reference software for sound and image processing is now Max/MSP. These environments have quickly grown up and incorporated modules including artificial intelligence techniques, especially artificial learning and many protocols to handle data coming from performers through various captors. As they are designed as graphical languages, these environments also became popular in the world of musicians using electronics but reluctant to classical programming for instance in C. So-called software "patches" tend to replace hardware devices. These environments propose kinds of metaphors of physics labs, where objects represent processes to perform on the signal (either digital, sound or image data) that are linked by virtual wires. Thus, Max implies a "patch-oriented programming": everything has to be conceived as signal processing problems.

Knowing that the Computing with Words (CW) paradigm has shown to fit very well all kinds of human expressions (feelings, perceptions, emotions, impressions, etc.),[14,15] we propose a library designed for Max that implements patches for CW. We believe the mechanisms of fuzzy logic as well as the way it implies users in computerized processes provide interesting research tracks in the field of performing arts.[3] Hence the idea is to implement usual fuzzy logic tools,[13] but also recent or new algorithms, such as the fuzzy 2-tuples,[11] or some particular partitioning method.[3] Indeed we believe these tools can be adapted for real performance, in cases where inputs from performers can drive scenographic elements in realtime.[4]

Several fuzzy libraries are available online and have been implemented

in many languages such as Java, C++, MATLAB. For example, *jFuzzyLogic*[12] offers custom membership functions; BoundedSum, Max, ProbOr, Sum, NormedSum, etc. *FuzzyJ* is a set of Java classes for handling fuzzy concepts and reasoning.[10] It is based on the FuzzyCLIPS extension to the CLIPS Expert System Shell. The *Free Fuzzy Logic Library*[7] is an open source fuzzy logic class library and API written in C++ from 2001 to 2003. It has been designed to be optimized for speed critical applications, such as video games. The *Fuzzy Toolbox*^{TM}has been written for MATLAB.[9] It implements classical fuzzy control concepts but offers only Mamdani and Sugeno inference systems.

Despite these numerous fuzzy library implementations, none of them is tailor-made for real time signal processing. However, several authors have proposed patches to enable the use of some AI techniques. As an example, in a recent work[5] Eigenfeldt describes methods of creating networked multi-agents within Max/MSP and also some fuzzy logic ratings for a drum ensemble. Less recently Elsea explains how the core concepts of fuzzy logic may be applied to problems common in music analysis and composition.[6] Some other patches were also proposed (e.g. Fuzzy Harmony patcher) but they are either too specific and don't allow for a real CW framework.

Therefore the fuzzy logic tools have to be reconsidered through this way of thinking and designing.

3. Implementing a Tool for Computing [performances] with Words in Max

Max/MSP has been designed to permit a minimal sol-fa representation of the signal. In particular it proposes to detect very simply the tone, the pitch, the onsets, etc. of a signal. But all high-level concepts such as melody, pizzicato, legato, staccato, etc. are hard to detect. Max aspires to map semantic concepts on signal processing while CW offers semantic concepts without any link toward signal processing. CW tools are easy to implement using variables, functions, methods in classical programming languages such as Java or C++, whereas in Max programming there are no variables and methods must be functionally grouped together into objects or patches.

The software library FuzzyLib for Max/MSP is currently available at `http://imtr.ircam.fr`. Three objects (Max sense) have been implemented: *lv1* standing for 'linguistic variable version 1' (see Figure 1) whose purpose is the fuzzification of a phenomenon represented by a numerical value; *gmpa1*, standing for 'generalized modus ponens application version 1', that receives fuzzy rules expressed as Max/MSP *messages*. All the clas-

sical implications have been implemented; an interface object named *rule-Composer* to help users to write fuzzy rules and avoid syntax errors. 'Not' operator (in addition to 'And' and 'Or') has also been implemented.

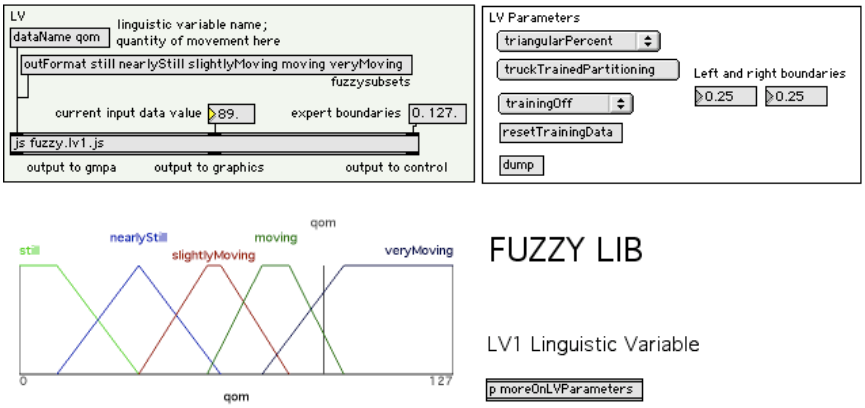

Fig. 1. A screenshot of the help patch of the 'lv' object.

4. FuzzyLib: example with a theatre play

FuzzyLib has been tested in the theatre play *Les petites absences* framework. The whole team and in particular stage director Marco Bataille-Testu have used the library before and during the show. Three sub-tasks have been identified: acquiring and qualifying the performer observation, and modeling semantically the human-to-computer interaction on stage.

First, the idea is to retrieve low- and high-level descriptors to qualify the performer gesture and voice. The video input is based on the realtime captation but it can also work on pre-recorded sequences to calibrate fuzzy subsets from training data (see object *lv1* above). The video descriptors provided by *cv.jit* objects (Computer Vision library) are for instance quantity of movement, contraction surface, silhouette orientation, etc. The audio descriptors retrieved thanks to *analyzer* object are for instance fundamental pitch, loudness, etc. Of course the instantaneous values shall not be used *as they are*, they are aggregated by relevant operators. These sets of aggregators are given as inputs of *lv* during both the learning phase and the live audience. A partitioning that linguistically qualify each aggregator is processed: this enables to qualify the performances and the performer's activity.

In *Les petites absences* fuzzy rules were mainly used to control a generator of crowd sounds — that would fit the behavior of the main actor — or to trigger some events. M. Bataille-Testu wrote by himself the fuzzy rules. The outcome is very positive since the play has been performed 25 times since December 2008 giving satisfying results. However, there are several interactive processes that cannot be specified using fuzzy rules: for instance, it is very difficult to program (with fuzzy rules) contrapuntic imitation of an instrumentist.

We propose several ways to use fuzzy rules in the field of performing arts. In the past we used them to recognize simple emotions in the performance of pre-defined scenes.[2] We are now more interested in *driving scenographies* thanks to fuzzy rules. As they enable to work out relationships between input linguistic variables (on the performer side) and output ones (on the electronic side), they constitute interesting ways of stating elements of interactive scenography. As the users may enter their own vocabulary, they provide original ways for artists to specify interactions for stage shows.

During our experiments one difficulty was the understanding of fuzzy rules. First of all, it is not obvious for a non-specialist that the result of a fuzzy implication will give a fuzzy subset and not a crisp value. But with the help of words, this point becomes less problematic. That is why the linguistic counterpart is so important, especially here. Second point is the implicit reciprocity in the rules stating that the reverse is true: e.g. 'if the movement is nearly still, then the frequency is high' may implicitly mean that 'if the movement is *not* nearly still, then the frequency is *not* high'. Third and last point is the choice of fuzzy implication: it is hard to imagine that, when the premise is false, the result may be always false (in the case of Mamdani-Larsen implications) and always true in all other cases.

5. Conclusions and Future Work

Our contribution in this paper is a thought about a library for CW concepts and the fully customizable library itself. It has been included into Max/MSP which is an environment for artists looking for intuitive programming. This research has been validated on stage in a theatre show that has already been performed 25 times. FuzzyLib has been useful both in the preparation of the show, enabling the stage director to be implied in the Max/MSP design without mathematical mapping definition, and in the performance itself, letting actors some flexibility.

Next step is to propose the version 2 of FuzzyLib that will include new functions to handle vectors directly and write fuzzy rules on aggregations

(standard deviation, minimum, maximum, etc.) on vectors instead of vectors themselves, e.g. "if $\min[X]$ is ...". Another point is to include temporal aspects in the fuzzy rules: e.g. "if $X[n] > X[n-1]$ then ...". This way we will be able to handle dynamic systems. This shall imply to handle fuzzy order relations.[13,14] We would also like to develop some applicative aspects such as the gesture following and recognition with Hidden Markov Models, as proposed by Bevilacqua *et al.*[1] at Ircam: an approach using the FuzzyLib would probably be relevant and helpful.

References

1. F. Bevilacqua, F. Guédy, E. Fléty, N. Leroy, and N. Schnell. Wireless sensor interface and gesture-follower for music pedagogy. In *Proc. of the Int. Conf. on New Interfaces for Musical Expression*, pages 124–129, 2007.
2. A. Bonardi and I. Truck. First steps towards a digital assistant for performers and stage directors. In *Proc. of the 3rd Int. Conf. on Sound and Music Computing*, pages 91–96, 2006.
3. A. Bonardi, I. Truck, and Akdag H. Building Fuzzy Rules in an Emotion Detector. In *Proc. of the 11th Int. Conf. on IPMU*, pages 540–546, 2006.
4. A. Camurri, A. Catorcini, C. Innocenti, and A. Massari. Music and multimedia knowledge representation and reasoning: the harp system. *Computer Music Journal*, 19(2):34–58, 1995.
5. A. Eigenfeldt. Drum Circle: Intelligent Agents in Max/MSP. In *Proc. of the International Computer Music Conference*, 2007.
6. P. Elsea. *Fuzzy Logic and Musical Decisions*. University of California, Santa Cruz, 1995. `http://arts.ucsc.edu/EMS/Music/research/FuzzyLogicTutor/FuzzyTut.html`.
7. FFLL. Free fuzzy logic library. `http://ffll.sourceforge.net/index.html`.
8. G. Fontaine. *Le décor d'opéra*. Editions Plume, Paris, 1998.
9. Fuzzy Toolbox. Fuzzy logic toolbox^{TM}2.2.9. MATLAB. `http://www.mathworks.com/products/fuzzylogic/`.
10. FuzzyJ. The NRC FuzzyJ Toolkit. National Research Council. `http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit2.html`.
11. F. Herrera and L. Martínez. A 2-tuple fuzzy linguistic representation model for computing with words. *IEEE Trans. Fuzzy Systems*, 8(6):746–752, 2000.
12. jFuzzyLogic. Open source fuzzy logic library and fcl language implementation. `http://jfuzzylogic.sourceforge.net/html/index.html`.
13. Kaufmann, A. *Introduction to the Theory of Fuzzy Subsets*. Academic Press, New York, 1975.
14. L.A. Zadeh. Similarity relations and fuzzy orderings. *Information Sciences*, 3:177–200, 1971.
15. L.A. Zadeh. From computing with numbers to computing with words: From manipulation of measurements to manipulation of perceptions. *Int. J. of Applied Math and Computer Science*, 12(3):307–324, 2002.