CONTROLLING A SOUND SYNTHESIZER USING TIMBRAL ATTRIBUTES

Antonio Poščić
Faculty of Electrical Engineering and Computing,
University of Zagreb, Croatia
antonio.poscic@fer.hr

Gordan Kreković
Faculty of Electrical Engineering and Computing,
University of Zagreb, Croatia
gordan.krekovic@fer.hr

ABSTRACT

In this paper we present the first step towards a novel approach to visual programming for sound and music applications. To make the creative process more intuitive, our concept enables musicians to use timbral attributes for controlling sound synthesis and processing. This way, musicians do not need to think in terms of signal processing, but can rely on natural descriptions instead. A special point of interest was mapping timbral attributes into synthesis parameters. We proposed a solution based on fuzzy logic which can be applied to different synthesizers. For a particular synthesizer, an audio expert can conveniently define mappings in form of IF-THEN rules. A prototype of the system was implemented in Pure Data and demonstrated with a subtractive synthesizer. A survey conducted among amateur musicians has shown that the system works according to their expectations, but descriptions based on timbral attributes are imprecise and dependent on subjective interpretation.

1. INTRODUCTION

The visual programming paradigm became very popular among computer musicians and multimedia artists in the last decade. One of the main reasons for the increasing acceptance of this approach is a faster learning curve compared to traditional textual programming [1]. Graphical representations of computer programs are closer to the way how humans mentally represent problems, so it is easier to understand and develop programs using the visual approach [2]. Many visual programming editors support direct manipulation of graphical objects which further helps users to perceive how their actions affect the program [3]. The visual programming paradigm also brings other psychological benefits which are particularly important for musicians and artists as they usually do not have strong backgrounds in programming [4].

Besides these benefits, there is one more factor specifically related to computer music, multimedia, and interactive art. In this domain, digital processing of audio signals, images, and videos is a fundamental part of every application. Many modern visual programming environments offer ready-to-use program elements which implement digital signal processing algorithms and facilitate integration with peripheral devices. In the context of music and sound processing, such elements are oscillators, filters, audio effects, score following algorithms, auto-tuners, etc. Using prepared elements, musicians and artists do not need to cope with low-level signal processing. Thanks to the general psychological benefits of the visual programming paradigm and visual programming environments designed to meet practical needs, visual programming is now widely recognized among computer musicians and multimedia artists.

In visual programming environments for sound and music processing, audio signals usually participate in the data flow. Pure Data, Max/MSP, Kyma, AudioMulch, and Reaktor are some of the most popular environments which rely on this paradigm. While visual programming based on signal flow ensures maximal flexibility, it forces the user to think about music and sound art in terms of signal processing. Musicians have to understand how certain program elements affect the audio signal and which parameter values should be chosen to achieve the desired sound quality.

With regards to these benefits of visual programming and to make the creative process more straightforward for musicians, within this research we present our vision and the foundations of a novel approach to visual programming for sound and music applications. One of the cornerstones of this concept relies on the notion that control over sound synthesizers, audio effects, and other elements for generating or modifying audio signals is established through timbral attributes. Instead of manipulating audio signals and parameters of program elements, musicians can focus on their musical ideas and realize them by describing timbral characteristics of a desired sound. For example, a musician can specify that the sound needs to be “metallic”, “bright”, and “harsh”. Besides the target timbre, it is also possible to define timbral changes through time. The focus of this paper is, thus, on the usage and transformation of timbral characteristics. The approach presented in this research will be later used as one of the main building blocks of the aforementioned innovative visual programming language and environment.

There are two main factors which make the described concept challenging. The first one is a lack of theoretical and notational support related to timbre [5]. While other characteristics such as pitch and rhythm have more formal notations, timbral attributes are not standardized.

Copyright: © 2013 Antonio Poščić, Gordan Kreković. This is an open-access article distributed under the terms of the Creative Commons Attribution License 3.0 Unported, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

They are meaningful to musicians, but not convenient as an input to computer systems.

The second big challenge is mapping timbral attributes into parameters of sound synthesizers or audio effects. Such relations are usually complex and ambiguous. Additionally, mappings should work for different synthesis techniques and different types of audio effects so they have to be adequately generic. Existing works include several attempts at synthesizing sound specified by timbral attributes. Miranda used a machine learning algorithm to induce relations between quasi-timbral attributes and synthesis parameters [6]. However, the available attributes were always associated with the structure of a sound synthesizer so Miranda's approach would not work for any synthesizer other than the one designed as a part of his system. A research conducted by Gounaropoulos and Johnson employed a neural network to learn mappings between timbral attributes and audio features of a sound characterized by such attributes. This research used the backward-propagation algorithm to control the synthesizer [7]. As the algorithm was specifically designed to work in the case when synthesis parameters are directly related to audio features, it can be only applied to additive synthesis. The problem of controlling synthesis parameters with timbral attributes was not sufficiently explored nor solved in such a way that the solution could be adapted to different synthesis techniques.

Since this is also one of the central problems in our concept of a visual programming language and environment based on attribute flow, as mentioned before, in this paper we primarily focused on that issue. We devised, implemented, and evaluated a novel approach for mapping timbral attributes into synthesis parameters using fuzzy logic. Such a solution can be applied to an arbitrary sound synthesizer. The concrete implementation and demonstration was done using the programming language C and the visual programming environment Pure Data. We developed an external (Pure Data plugin) which enabled using a fuzzy logic library within Pure Data and demonstrated the solution on a subtractive synthesizer.

2. ATTRIBUTE-BASED VISUAL PROGRAMMING FOR MUSICAL COMPOSITION

The encompassing approach and concept presented in this paper are based on the notion that combining intuitive inputs (such as timbral characteristics) with visual programming elements and time-dependent control flow will enable musicians and other users to innovate their approaches towards music, sound creation, and sound modeling. Similar concepts based on a fusion of different paradigms can be observed throughout various works and designs in the field. One notable example is the sound design language Kyma [8] which enables the manipulation of sound objects in the domain of time.

Our concept defines that timbral attributes should be used in a way that can be usually found in the field of dataflow languages while also allowing time-based control of sound flows. This duality of our approach, not found in similar works, results in a visual programming language and environment relying on two paradigms [9]: icon based [10] and diagram based [11] visual programming. The icon based programming portion is linked with defining and selecting the timbral attributes through various possible user interfaces. On the other hand, the diagram based side of the concept is related to the links and interdependence between various manipulation objects such as synthesizers, VST plugins, etc. The time-bound connection between portions of the target sequence of sounds is established in a manner that resembles audio editing software such as Audacity [12].

Considering the aforementioned concepts, it's important to stress that attribute-based flow has not been selected by accident but it's rather a design choice made to enhance symbiosis with the targeted visual programming paradigm. The paradigm thus contains user interface elements derived from tools such as the aforementioned Audacity as well as elements belonging to visual programming languages such as Pure Data. By combining these traits, we enable the users to efficiently explore the possible synthesized sounds both in the time domain and in the different domains of sound characteristics. Using concepts that are usually present in diagram and dataflow languages, such as the possibility to connect blocks that manipulate the attribute flow, the user will be able to architect sound sequences and define links between sounds. On the other hand, the user will have means to change and adapt the individual generated sounds by directly influencing the behavior of sound synthesizing blocks. The approach based on fuzzy logic that is demonstrated in this article should be seen as one of the possible methods encapsulated in these blocks. Our preliminary research has shown that attribute flow is best suited to achieve these objectives and desired characteristics. These ideas and implementations will be further explored in future works.

One of the crucial characteristics of the system is also the ability to include a variety of different approaches with regards to mapping timbral attributes into synthesis parameters. The possible methods, beside the fuzzy logic method described in this research, include neural networks, classifier cascades, regression tree analysis, etc.

3. PROOF OF CONCEPT

The first and very important step towards the concept of visual programming based on attribute flow was to establish mappings between timbral attributes and synthesis parameters. To solve this problem, we suggest an approach which uses an expert system based on fuzzy logic. Within this research we implemented that solution and tested it with a subtractive sound synthesizer.

3.1 Mapping timbral attributes into synthesis parameters

Previous attempts to synthesize sounds from their timbral descriptions relied on machine learning algorithms [6, 7]. The idea was to induce relations between timbral attributes and synthesis parameters by learning from examples of synthesized sounds. Such a problem is hard to solve for a general case, since there can be more than one solu-

tion in the space of synthesis parameters for a certain sound quality. Relations between timbral attributes and synthesis parameters are therefore ambiguous. To formulate the problem so that it can be solved with known machine learning techniques, both previous researches were limited to one specific sound synthesizer. As described in the first chapter, authors used a priori knowledge about employed synthesizers and adapted algorithms to work with them.

To find a solution which could be applied to sound synthesizers with different structures, we decided to use an expert system based on fuzzy logic instead of a machine learning technique. Fuzzy logic is a form of probabilistic logic derived from the mathematical branch of fuzzy sets [13]. Compared to the traditional two-value logic where variables can be either true or false, fuzzy logic supports the concept of partial truth so variables can have a truth value that ranges between 0 and 1. This concept makes fuzzy logic convenient for quantifying imprecise information and making decisions based on incomplete data [14]. Furthermore, facts and rules can be described using linguistic terms which make the concept closer to human reasoning.

Linguistic variables are subjective and context-dependent variables whose values are words. For example, if the timbral attribute "warmth" is observed in the role of a linguistic variable, its values could be "very high", "high", "moderate", "low", and "very low" as shown in Figure 1. Inputs in a fuzzy logic system are usually numeric so it is necessary to convert these numeric values into linguistic terms. The linguistic terms can be considered fuzzy sets since an input value can have partial membership in more sets at the same time. For example, if the attribute "warmth" has the numeric value 0.85, it is situated between "very high" and "high" according to Figure 1. Therefore, the value belongs to both sets but not by the same degree. Fuzzy sets can use different types of membership functions such as triangular, trapezoidal, bell-shaped, and sigmoid functions.

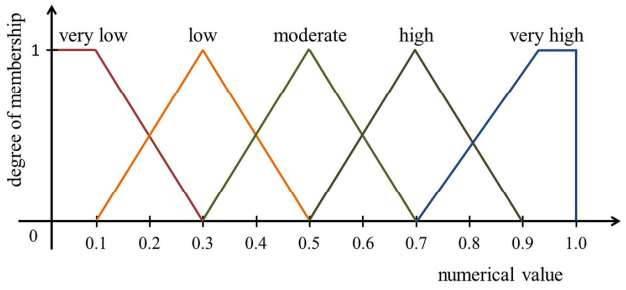

Figure 1. Fuzzy membership functions for the attribute "warmth".

A model for making decisions based on fuzzy logic uses a rule set defined by linguistic variables. Typically, fuzzy rules are specified in the form of IF-THEN statements:

IF (x_1 IS S_1) AND/OR ..., (x_n IS S_n) THEN y IS T,

where x_i represents input fuzzy variables, y is the output variable, while S_n and T stand for input and output fuzzy sets. The first step of applying the model is to convert input variables into fuzzy logic variables using membership functions. Subsequently, output variables are calculated by evaluating the rules. In most applications, outputs have to be numerical values so fuzzy output variables must be defuzzified. Algorithms used for evaluating the rules and for the defuzzification process are explained in [14, 15].

Fuzzy rule sets are appropriate for representing expert knowledge of a certain domain. As the rules have a simple form, they can be conveniently written, discussed, and tuned by human experts. Systems based on fuzzy logic have been used in many different fields such as engineering, economics, finance, geology, meteorology, and sociology. Various problems from the musical domain were also approached using fuzzy logic. It has been employed for evaluating computer music [16], recognizing rhythmic structures [17], coding of musical gestures for interactive live performances [18], analyzing the emotional expression in music performance and body motion [19], mapping visual information into aural information and vice-versa [20], sound synthesis [21], and several other applications.

For setting synthesis parameters, Hamandicharef and Ifeachor employed an expert system based on fuzzy logic [22]. The purpose of their system was to find such synthesis parameters so that the synthesized sound mimics the target sound. The inputs of the expert system were audio features extracted from the target sound, while the outputs were parameters for the sound synthesizer. Audio experts were involved in building the fuzzy model and specifying relations between audio features and synthesis parameters.

We believe that having experts define rules is also beneficial in our case. Namely, for each synthesizer intended to be controlled using timbral attributes, a fuzzy model has to be defined once. Since the model is stored outside the program code, there is no need for programming, so the model can be easily defined and tuned by sound synthesis experts. In practice, such experts could be synthesizer manufacturers or users who thoroughly understand the architecture of a specific synthesizer. Once the model is ready, other users can control the synthesizer by adjusting timbral attributes without writing or changing the fuzzy model.

This way, the expert system based on fuzzy logic can be used for sound synthesizers with various structures. The only requirement to adapt the system to work with a different synthesizer is to write a new fuzzy model. On the other hand, the previous solutions based on machine learning algorithms were not capable of such adaptation. Those solutions relied on a priori knowledge about the synthesizer structures, so training machine learning algorithms to work with different synthesizers was impossible.

We expect an expert system based on fuzzy logic to work well with synthesis techniques for which the relations between timbral attributes and synthesis parameters are continuous or semi-continuous. Most of the popular synthesis techniques, such as subtractive and additive synthesis, satisfy the criterion. On the other hand, frequency modulation is not one of the supported synthesis techniques since the ratio of modulator and carrier fre-

quencies is in a very complex relation with the perception of harmonicity. For that reason, we will limit this research to subtractive synthesis and extend it to other synthesis techniques in the future.

The selection of attributes used to describe the desired sound in our system was taken from [7]. Those attributes are: bright, warm, harsh, thick, metallic, woody, hit, plucked, and constant amplitude. The set of timbral attributes is not supposed to be orthogonal and it should instead only serve as an intuitive vocabulary for defining target sounds. In the sound description, an absolute value between 0 and 1 is assigned to each attribute. A value of 0 means that the particular quality is not presented in the sound, while the value 1 indicates that the quality is very prominent.

3.2 Implementation

Within this research we have developed a system for mapping timbral attributes to synthesis parameters which can be used in the Pure Data visual programming environment. To create an interface between Pure Data and a fuzzy logic library, we have implemented an external Pure Data component. It accepts a list of timbral attributes as input and calculates defuzzified outputs for the variables declared in the Fuzzy Control Language file. The Fuzzy Control Language (FCL) is a standardized (International Electrotechnical Commission standard, IEC 61131-7) language used to define and implement fuzzy logic models.

A similar implementation of a Pure Data external exists which is based on the *libfuzzy* library and the Fuzzy Inference System (FIS) notation [23]. However, the FIS notation is inferior both in usability and flexibility when compared to FCL. One example of the characteristics which make FIS less ergonomic is the requirement for each rule to be written using variable indices. This kind of deficiency could prove to be an insurmountable obstacle for typical users such as musicians when creating a large rule set.

The fuzzy logic implementation described in this paper relies on the *jFuzzyLogic* library [24]. Since Pure Data externals are natively written in C, a number of different C/C++ fuzzy logic libraries had been evaluated before we chose the *jFuzzyLogic* library. For example, the aforementioned *libfuzzy* library presents a valid fuzzy logic implementation in cases when the FIS input format is acceptable. None of the available open source C/C++ libraries were adequate due to obsolescence and improper or incomplete support for the Fuzzy Controller Language.

Since *jFuzzyLogic* was written in Java, wrapper functions that rely on Java Native Interface calls [25] have been implemented. These functions serve as glue code between the main functions of the external written in C and the *jFuzzyLogic* library. Proper error handling is also provided through these wrapper functions. To improve performance, a caching system based on the *sglib* library [26] has been implemented to reduce the number of output recalculations.

For demonstration and evaluation of the system we used a simple subtractive synthesizer with one oscillator and sub-oscillator, a noise generator, two filters (low-pass and band-pass), an amplitude envelope generator, two low-frequency oscillators, and a chorus effect. The block diagram of the synthesizer is shown in Figure 2.

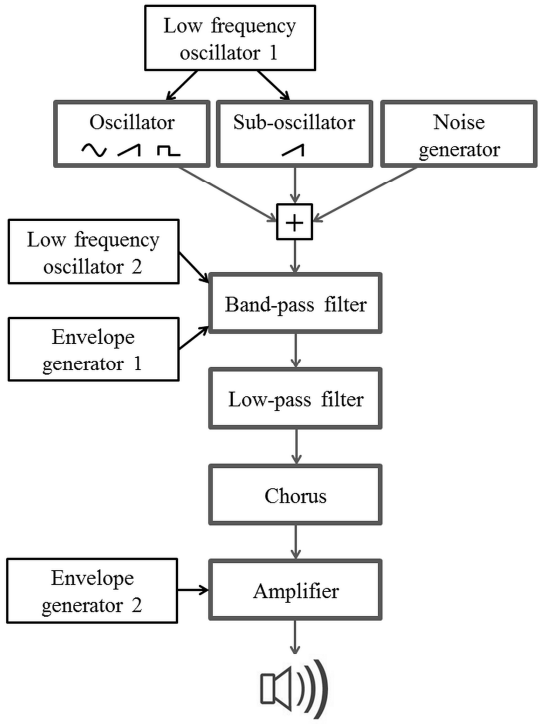

Figure 2. Block diagram of the subtractive synthesizer used in the experiments.

This is obviously a simple synthesizer which is not completely capable of producing sound qualities for all chosen timbral attributes as one might expect. However, it should not be considered a limitation, because the main goal of our system is to mimic a human expert and do the best with a given sound synthesizer regardless of its structure and capabilities. For the purposes of the evaluation, we implemented the synthesizer as a Pure Data patch and connected it to our external object as shown in Figure 3.

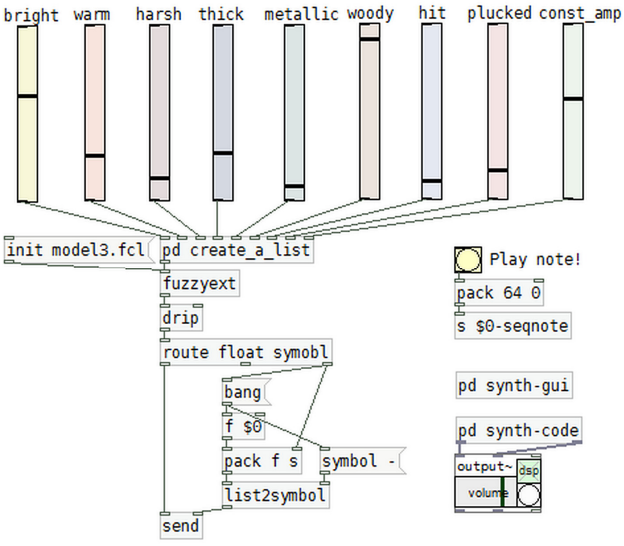

Figure 3. Example of using the external object within a Pure Data patch. The external object for fuzzy logic is called *fuzzext*.

The fuzzy logic model for mapping timbral attributes into synthesis parameters was defined in the FCL language. Inputs of the model are 9 timbral attributes ranging from 0 to 1 and there are 24 outputs with different ranges representing synthesis parameters. For the fuzzification and defuzzification processes, we defined triangular and trapezoidal functions specifically for each input and output variable. As the defuzzification technique, the model uses center of gravity. The rule set for calculating synthesis parameters consists of 85 rules in IF-THEN form. Some examples of the rules are as follows:

```
IF harsh IS little THEN filterlfo_r IS small;

IF plucked IS very_prominent OR hit IS prominent OR
hit IS very_prominent THEN volume_s IS very_small;

IF warm IS prominent OR warm IS very_prominent THEN
filterfreq_r IS moderate;

IF warm IS prominent OR warm IS very_prominent THEN
osc1type_r IS square;

IF woody IS very_prominent OR woody IS prominent OR
woody IS moderate THEN osc1type IS square;
```

4. EVALUATION

The purpose of the evaluation was twofold: to test the functionality of our system and to assess to what extent the chosen timbral attributes are appropriate for describing sounds. The evaluation was conducted among 6 amateur musicians who were asked to manually set synthesis parameters for 5 given sound descriptions. These descriptions were formed in the same way as the inputs in our system. For example:

> bright 0.8, warm 0.6, harsh 0.1, thick 0.2, metallic 0.1, woody 0.7, hit 0.1, plucked 0.1, constant amplitude 0.6

The participants did not receive any further explanations regarding the attributes so they had to interpret the given descriptions entirely by themselves. To compare the results of manual parameter manipulation with the results obtained algorithmically, we used the same descriptions as inputs in our system and generated the sounds.

After the participants finished their tasks of manual sound design, they received the sounds generated by our system and a survey. In the first set of questions they were asked to evaluate how the sounds synthesized by our system fit the sound descriptions. The second set of questions regarded the similarity between their sounds and the sounds generated using our system. All these questions were of rating scale type with 5 available options. For the first set of questions the scale included a range from very poor (1) to very well (5), while the second set included options ranging from very different (1) to very similar (5). Finally, the third set consisted of the following general questions:

1. How clear were the given descriptions based on attributes? (1 - very unclear, 5 - very clear)
2. How difficult was setting the parameters manually? (1 - very easy, 5 - very difficult)
3. How helpful the system for automatic synthesis from timbral attributes can be helpful for musicians? (1 - very little, 5 - very much)

5. RESULTS

The average grade for the questions concerning how the generated sounds met the given descriptions was 3.9 and the median grade was 4. These statistical values were calculated taking into consideration all 6 participants and all 5 sounds. The best rated sound with the prominent "plucked" attribute had an average grade of 4.2, while all other sounds had averages 4.0 or below. The questions regarding perceptual similarity to their sounds were graded by the participants with an average of 2.7. The median was 4. Distribution of all grades is shown in Figure 4.

The average grades on last three questions were 3.7, 3.2, and 3.9 respectively for clarity of such descriptions, difficulty of manual parameter setting, and potential usefulness of our approach.

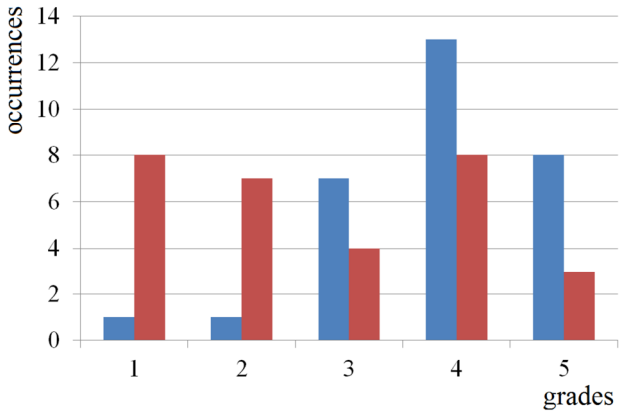

Figure 4. For each grade from 1 to 5 this chart shows how many times that grade appeared in the answers from all participant and for all sounds. The blue bars represent grades for achieving given sound descriptions, whilst the red ones represent grades for perceptual similarity to participants' sounds.

6. DISCUSSION

The results have shown that descriptions based on timbral attributes are somewhat imprecise for synthesizing exact sounds. The sounds that were created manually are perceptually different from automatically generated examples, but the participants were still quite satisfied with the results. To overcome this problem, our system could be improved to generate more than one option for any given description, thus allowing musicians to choose the most accurate instance of a sound.

The answers on the last three questions suggest that the problem of synthesizing sounds from timbral attributes is relevant to musicians and that our approach could be viable.

7. CONCLUSION

Parameters of a sound synthesizer can be successfully controlled from timbral attributes using an expert system based on fuzzy logic. For different types of synthesizers, fuzzy models can be defined by audio experts and later used without adaptation. Using an example based on a subtractive synthesizer, we have shown that the system

satisfies expectations of target users. The main problem of this approach is the lack of unified and strict relations between the chosen timbral attributes and the general perception of the sound. For that reason, future research could examine other sets of timbral attributes and improve the system so that it can generate several options for a given description.

The results of this research are generally encouraging with regards to our intention to develop a visual programming environment for music and sound processing based on attribute flow.

8. REFERENCES

[1] K.N. Whitley, "Visual programming languages and the empirical evidence for and against," *Journal of Visual Languages and Computing*, vol. 8, no. 1, pp. 109–142, 1997.

[2] O. Clarisse and S.-K. Chang, "VICON: A Visual Icon Manager" in *Visual Languages*. Plenum Press, 1986.

[3] B. Shneiderman, "Direct Manipulation: A Step Beyond Programming Languages," *IEEE Computer*, vol. 16, no. 8, pp. 57–69, 1983.

[4] D.C. Smith, *Pygmalion: A Computer Program to Model and Stimulate Creative Thought*. Birkhaüser-Verlag, 1977.

[5] T. Wishart, *On Sonic Art*. Taylor & Francis Group, 1996.

[6] E. Miranda, "An Artificial Intelligence Approach to Sound Design," *Computer Music Journal*, vol. 19, no. 2, pp 59–75, 1995.

[7] A. Gounaropoulos and C. Johnson, "Synthesising Timbres and Timbre Changes from Adjectives/ Adverbs" in *Applications of Evolutionary Computing*. Springer-Verlag, 2006.

[8] C. Scaletti, "Composing sound objects in Kyma," *Perspectives of New Music*, pp. 42–69, 1989.

[9] E.J. Golin and S.P. Reiss, "The Specification of Visual Language Syntax," *Journal of Visual Languages and Computing*, vol. 1, no. 2, pp. 141–157, 1990.

[10] C. Frasson and M. Erradi, "Principles of an icons-based language," in *Proc. of the 1986 ACM SIGMOD int. conf. on Management of data*, New York, 1986, pp. 144–152.

[11] G. Engels and R. Heckel, "From trees to graphs: defining the semantics of diagram languages with graph transformation" in *ICALP Satellite Workshops*, Geneva, 2000, pp. 373–382.

[12] The Audacity Team. (2012). Audacity: Free Audio Editor and Recorder [Online]. Available: http://audacity.sourceforge.net/

[13] L.A. Zadeh, "Fuzzy sets," *Information and Control 8*, pp. 338–353, 1965.

[14] B. Kosko, *Fuzzy Thinking. The new science of fuzzy logic*. Hyperion, 1993.

[15] G.J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall PTR, 1995.

[16] M. Miličević, "Aesthetics of Designing an Adaptive Fuzzy System for the Evaluation of the Computer Music" in *Proc. of the Int. Computer Music Conf.*, Beijing, 1999, pp. 538–541.

[17] T. Weyde, "Grouping, Similarity and the Recognition of Rhythmic Structure" in *Proc. of the Int. Computer Music Conf.*, Havana, 2001.

[18] N. Orio and C. De Piro, "Controlled Refractions: A Two Levels Coding of Musical Gestures for Interactive Live Performances" in *Proc. of Int. Computer Music Conf.*, Ann Arbor, 1998.

[19] A. Friberg, "A Fuzzy Analyzer of Emotional Expression in Music Performance and Body Motion" in *Proc. of the Music and Music Science*, Stockholm, 2004.

[20] R. Cádiz, "A Fuzzy-Logic Mapper for Audiovisual Media," *Computer Music Journal*, vol. 30, no. 1, pp. 67–82, 2006.

[21] E. Miranda and A.J. Maia, "Granular Synthesis of Sounds Through Markov Chains" in *Proc. of the Int. Computer Music Conf.*, Barcelona, 2005.

[22] B. Hamandicharef and E. Ifeachor, "Intelligent and Perceptual-Based Approach to Musical Instruments Sound Design", *Expert Systems and Applications*, vol. 39, no. 7, pp. 6476–6484, 2012.

[23] R. Cádiz and G. Kendall, "Fuzzy Logic Control Tool Kit: Real-Time Fuzzy Control for Max/MSP and Pd" in *Proc. of the Int. Computer Music Conf.*, New Orleans, 2006.

[24] P. Cingolani and J. Alcalá Fernández, "jFuzzyLogic: A Robust and Flexible Fuzzy-Logic Inference System Language Implementation" in *Proc. of the 2012 IEEE Int. Conf. on Fuzzy Systems*, Brisbane, 2012, pp 1–8.

[25] Oracle. (2011). Java Native Interface [Online]. Available: http://docs.oracle.com/javase/6/docs/technotes/guides/jni/

[26] M. Vittek, P. Borovansky, and P.-E. Moreau, "A Simple Generic Library for C, in Reuse of Off-the-Shelf Components" in *Proc. of the 9th Int. Conf. on Software Reuse*, Turin, 2006, pp. 423–426.