An experimental SDIF-sampler in Max/MSP

Vincent V. Rioux, Manuel Poletti

submitted to International Computer Music Conference ICMC, Göteborg, Sweden, 16-21 september 2002

An experimental SDIF-sampler in Max/MSP

Vincent Rioux, Manuel Poletti

IRCAM - CENTRE GEORGES POMPIDOU
email: rioux@ircam.fr

Abstract

Samplers are traditionally distinguished from synthesizers according to their input data: raw audio for samplers or model-based representation for synthesizers. We would like to introduce here a somewhat intermediary concept by taking advantage of the remarkable flexibility and power of a combined use of a real-time environment such as Max/MSP and an audio-signal model description standard such as SDIF (Sound Description Interchange Format).
In short, our SDIF-sampler uses SDIF-files in a similar manner as a sampler uses audio-files.
We will review the different stages implied in the construction of such a sampler namely analysis, re/synthesis and user-interface design.
This concept though still in an experimental phase has been tested with a realistic set-up.

1. Introduction

We started to work on this sampler concept during the design phase of a prototype that could be used in conjunction with an existing acoustic pipe organ. The requirements of this system are reflected in three desired modes of use:

- a rehearsal mode: a sampled copy of the organ allows simulating a performance.
- a voicing mode: the sound quality of the sampled and/or virtual registers can be precisely tuned.
- a performance mode: real-time transformations (filtering, spatialization) can be applied.

Hence this project should combine signal processing, real-time and spatialization algorithms while keeping a good coherence with the concept of registers.

In the following we will describe how a prototype fulfilling the previous requirements was built. First we will present an automatic procedure suitable for the analysis of a set of audio samples in order to produce their model equivalents represented in SDIF. Then, we will show how the synthesis part and the user-interface were implemented in the Max/MSP environment.

2. Analysis

The analysis procedure is rather classic. It involves a pitch extraction followed by an estimation of time-varying sinusoids. The residual part obtained by subtraction of the sinusoidal part is finally approximated by a source-filter model. Our main concern here was to choose and use algorithms requiring as few parameters as possible while staying robust, regardless of effort expanded, thus possibly accepting high computation costs.

Most of the computations were done using Matlab and resulting output parameters were stored in SDIF files using the Ircam-SDIF library and the Matlab API (described in Schwarz and Wright 2000).

2.1. Pitch extraction

In order to automatically adjust parameters for subsequent analyses and tuning processes, pitches of all available samples need to be computed. A large number of pitch estimators are available in the literature. We used the Yin algorithm for its robustness and quality systematically tested over a large speech corpus (de Cheveigné and Kawahara 2002).

2.2. Sinusoidal model

In the case of a pipe organ, sounds are mostly harmonic. However some deviations exist and account for substantial level of random fluctuations (unstable wind pressure and turbulences) (Fabre 2000 and Verge, Hirschberg, and Caussé 1997) complex attack transients (Castellengo 1999) and combination of several sources (mixtures).

Estimation of time-varying sinusoids is indeed not that straightforward and requires above all to carefully treat the particular problem of large fundamental frequency intervals (from C1-32Hz to F#7-6KHz). This estimation was achieved through the use of an adaptive method where matched sinusoids are modeled by a spline function (Röbel, 2001). This method proved to be robust though being computationally expensive.

2.3. Source/filter model

In a first approximation, we modeled the residual part as a time-fixed filter excited by white noise. Triplets of Q factor, amplitude and frequency were used as parameters of the corresponding spectral envelope.

3. Synthesis in Max/MSP

The implementation of the synthesis engine was realized in the Max/MSP graphical programming environment, taking advantage of the existence of objects specifically constructed to read SDIF files in real-time applications (Wright *et al* 1999b).

In order to keep some coherence with the concept of registers, we organized samples and their SDIF equivalent in corresponding directories (or banks). Each register is associated with a text file in which each SDIF-sample receives a set of parameters. An example of such a set of parameters is presented in the table below. Each register parameter file is automatically generated using *Python* scripting-language (www.python.org).

30 F#1 **1000 1382** 30 0.0 **127 64** 01-F#1-fluteHarmo8.add 01-F#1-fluteHarmo8.res **-90 2**;

midipitch key-symbol **freeze-point duration** original-pitch(midi) transposition(midicents) **gain balance-H/N** ADDname RESname **azimuth elevation**

Several types of parameters appear. First, we find parameters correspond to the actual note. Then timing parameters like duration (*1382ms*) are presented, including a "freeze" point (*1000ms*) which corresponds to a simplified loop point, setting the parameters of the bank of oscillators to fixed values. Gains (127 64) parameters include a simple gain and a balance between the harmonic and residual part. Pitch parameters allow mapping a single analyzed sample to several neighboring notes. The names of the SDIF-files which contain the proper synthesis parameters are also passed as parameters (*ADDname* for the additive part and *RESname* for the resonant part). Finally, we find parameters azimuth and elevation allowing the control of spatial diffusion.

3.1. Central patch

Max/MSP graphical programming language (Puckette 1991) is organized in patches, which can be seen as functional modules exchanging audio or control signals (consult www.cycling74.com for a full description of what Max can do). Figure 1 shows such a module, the central patch of our SDIF-sampler. It receives inputs from for example a MIDI-keyboard (in1) then triggers the inspection of the required SDIF-files, builds list of sinusoidal (additive) and residual (resonant) parameters and forwards these parameters to a signal-processing engine.

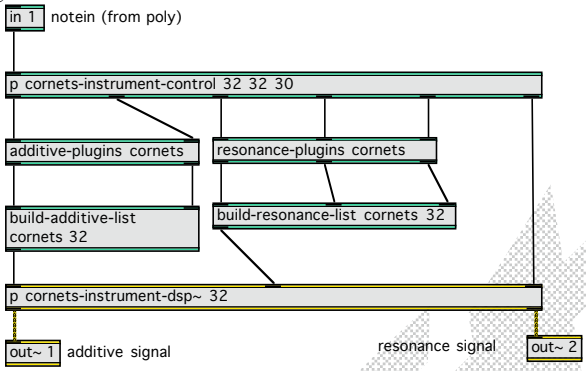

Figure 1. A "register" or "instrument" patch.

3.2. SDIF-buffers and tuples

In figure1, the "*cornets-instrument-control*" module dispatches parameters read from SDIF-files to the synthesis engine at specific time instants. For each register a number of <*sdif-buffer*> are allocated which point to specific SDIF-files (see figure 2 below). Figure2 shows how from the specification of a time instant parameter and a transposition parameter, <*SDIF-tuples*> objects are called in order to produce two lists of frequencies and amplitudes in the case of the additive synthesis part.

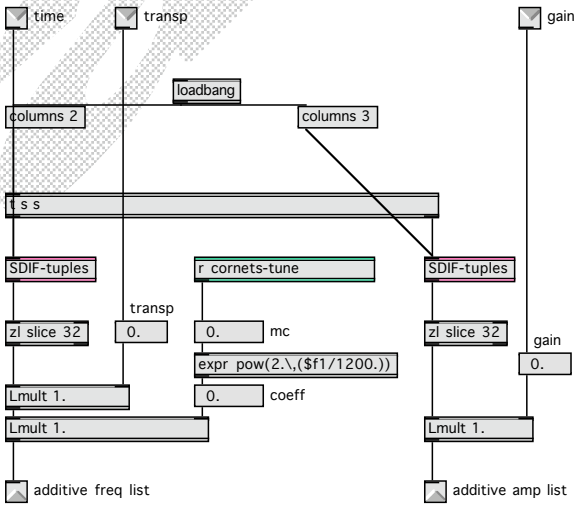

Figure 2. Real-time control of synthesis. Inputs (time, transposition factor and gain) trigger an output from each <*SDIF-tuples*> associated with an <*SDIF-buffer*> (Wright *et al* 1999b).

3.3. Synthesis engine

The real-time signal-processing engine is fed with these lists of parameters.

A number of patches are concurrent for the synthesis of additive and resonator parts of the signal. We chose to use <*oscbank~*> (from Cycling74) for the synthesis of the sinusoidal additive part and <*resonators~*> for the residual part (from CNMAT, see Jehan, Freed and Dudas 1999).

Moreover we used the following objects which do not explicitly belong to the Max/MSP library: <*Lobjects*> by Peter Elsea, <*List ops*> by James Mc Cartney, <*Vector ops*> by Laurent Cerveau and <*VT objects*> by Adrien Lefèvre.

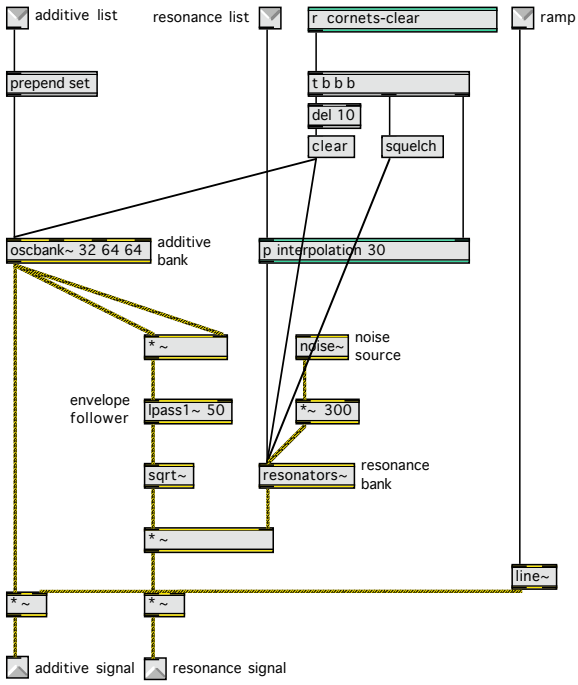

Figure 3. Real-time synthesis of the sinusoidal (additive) and residual (resonant) part.

3.4. Plugins architecture

The interest of an SDIF-sampler lies in the fact that a real-time precise control of all previously discussed parameters is possible. This control allows of course modification, filtering and combination of these parameters. As can be seen in Figure 4, Max/MSP offers an elegant way to do so.

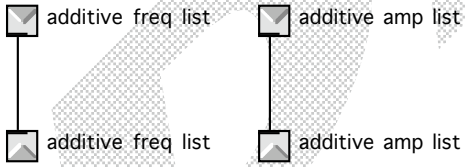

Figure 4. All parameters read from any *SDIF-tuple* are accessible in real-time and can be easily rearranged (here the simplest bypass case for the additive synthesis part, leaving imagination open...).

4. User-interface

On top of all these modules, we designed a simple interface which allows the user to call some registers and modify globally its parameters as can be seen on figure 5.

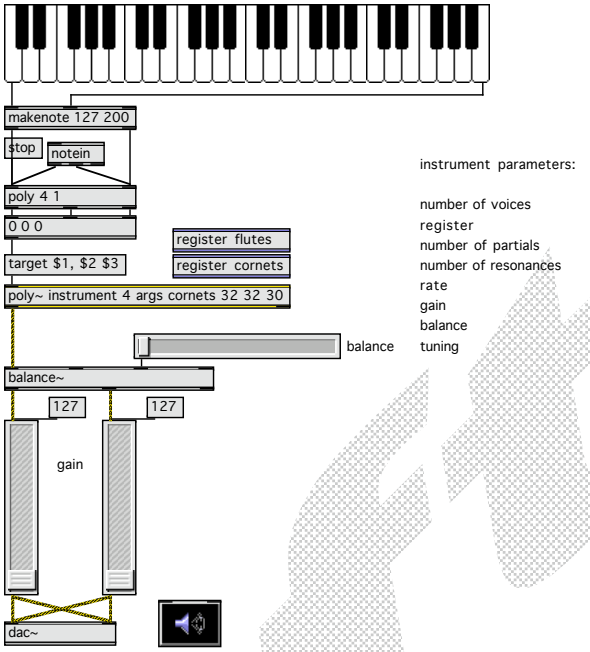

Figure 5. Simple user-interface of the SDIF-sampler.

5. Experiment

At this stage of development the system is limited to one register at a time with up to 12 voices of polyphony. These performances were tested on two G3 macintosh running Max4. We were hence able to launch an SDIF-sampler plus a spatialization module (run by the second computer) with no noticeable latency between key actions and electro-acoustic response.

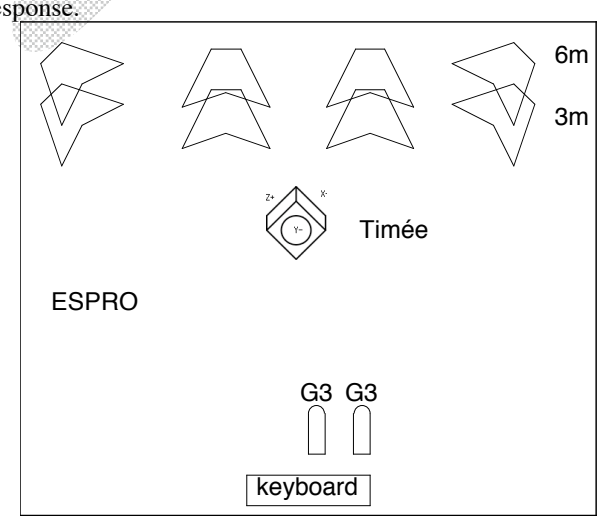

Figure 6. Disposition (not to scale) of loudspeakers for our experiments at Ircam (espace de projection). A matrix of 8 speakers is composed of two rows placed at 3 and 6 meters respectively. A "Timée" set-up (multispeaker) was also tested.

Two forms of spatialisations were tried out. We first used a classical technique involving a pan pot of a network of loudspeakers plus some reverberation or room effect modelization in order to model variation

of sources positions. Secondly, we used a 3D array of loudspeakers (Timée) (Misdariis and *al*. 2001a, Warusfel and Misdariis 2001b) which allows a control of directivity patterns thanks to a set of Max/MSP patches merged under the name "Spat~" (Jot and Warusfel 1995a and 1995b, Jot 1997).

6. Perspectives

A number of further development is already expected, including:

- scriptability of the sampler via MIDI
- integration of layers or pre-merged register files
- exploration of plugins
- integration of faster synthesis algorithms like FFT-1 (Freed, Rodet, Depalle 1992)
- voicing interface of the electronic part

Considering the timbral complexity of any acoustic instrument (and a fortiori of an acoustic pipe organ) we don't expect anybody to use this kind of system in order to imitate and replace but rather to be able to expand and create.

7. Acknowledgments

Many thanks go to Matt Wright, Xavier Rodet, Olivier Warusfel, Adrien Lefèvre, Nicolas Misdariis, Serge Lemouton, Diemo Schwarz, Thomas Hélie and Magalie Deschamps for their contributions to this work.

This work was funded by the Palais des Beaux-Arts, Brusells, Belgium, for a project involving the reconstruction of an acoustic organ with added electronics: "Projet de grand orgue, avec intégration de lutherie électronique".

References

Castellengo, M. 1999. “Acoustical Analysis of Initial Transients in Flute-like Instruments.” *Acta acustica*, **85** (3), 387-400.

de Cheveigné, A., Kawahara, H. 2002. "YIN, a fundamental frequency estimator for speech and music." in press *J. Acoust. Soc. Am.*

Fabre,B. 2000. “Physical Modeling of Flue Instruments: A Review of Lumped Models.” *Acustica*, **86**(4), 599-610.

Freed, A., Rodet, X., Depalle, Ph. 1992. "Synthesis and Control of Hundreds of Sinusoidal Partials on a Desktop Computer without Custom Hardware", *proceedings of the ICSPAT*, San José, USA.

Jehan,T., Freed, A. and R. Dudas 1999. "Musical Applications of New Filter Extensions to Max/MSP", *proceedings of the ICMC*, Beijing, China.

Jot, J.M., Warusfel, O. 1995a "Spat~: a spatial processor for musicians and sound engineers", *Proc. CIARM'95* Conference, Ferrara (Italie), Mai 1995.

Jot, J.M., Warusfel, O. 1995b "A real-time spatial sound processor for music and virtual reality applications", *Proc. ICMC'95 Conference*, Banff (Canada), Septembre 1995.

Jot, J.M., 1997. "Efficient models for distance and reverberation rendering in computer music and virtual audio reality", ICMC 97, septembre 1997.

Misdariis, N., Nicolas, F., Warusfel O., Caussé R. 2001a. "Radiation control on multi-loudspeaker device : La Timée". *XX th International Computer Music Conference* (ICMC) - septembre 2001.

Puckette, M. 1991. "Combining Event and Signal Processing in the MAX Graphical Programming Environment." *Computer Music Journal* 15(3): 68-77.

Röbel, A. 2001. "Adaptive additive synthesis using spline based parameter trajectory models." *Proceedings of the International Computer Music Conference*, Havanna, 2001

Schwarz, D. & Wright, M. 2000 "Extensions and Applications of the SDIF Sound Description Interchange Format." *Proceedings of the International Computer Music Conference*, Berlin 2000.

Verge, M.P., Hirschberg, A., and Caussé, R. 1997 “Sound Production in Recorderlike Instruments II: A Simulation Model.” *Journal of the Acoustical Society of America*, 101(5), 2925-2939.

Warusfel O., Misdariis N. 2001b "Directivity synthesis with a 3D array of loudspeakers - Application for stage performance". *Proc. of the Digital Audio Effects Conf. 2001* (DAFx), décembre 2001.

Wright, M., Chaudhary, A., Freed, A., Khoury, S., & Wessel, D. 1999a. "Audio Applications of the Sound Description Interchange Format Standard." In *AES 107th convention.*

Wright, M., Dudas, R., Khoury, S., Wang, R., & Zicarelli, D. 1999b. "Supporting the Sound Description Interchange Format Standard in the Max/MSP Environment." *Proceedings of the International Computer Music Conference*, Peking 1999.