OMaxist Dialectics: Capturing, Visualizing and Expanding Improvisations

Benjamin Lévy
STMS Lab
IRCAM, CNRS, UMPC
1, place Igor Stravinsky
75004 Paris
Benjamin.Levy@ircam.fr

Georges Bloch
CNSMDP & IRCAM
209 av. Jean Jaurès
75019 Paris
gbloch@cnsmdp.fr

Gérard Assayag
STMS Lab
IRCAM, CNRS, UMPC
1, place Igor Stravinsky
75004 Paris
Gerard.Assayag@ircam.fr

ABSTRACT

OMax is an improvisation software based on a graph representation encoding the pattern repetitions and structures of a sequence, built incrementally and in real-time from a live Midi or Audio source. We present in this paper a totally rewritten version of the software. The new design leads to refine the spectral *listening* of OMax and to consider different methods to build the symbolic alphabet labeling our symbolic units. The very modular and versatile architecture makes possible new musical configurations and we tried the software with different styles and musical situations. A novel visualization is proposed, which displays the current state of the learnt knowledge and allows to notice, both on the fly and a posteriori, points of musical interest and higher level structures.

Keywords

OMax, Improvisation, Machine Learning, Machine Listening, Visualization, Sequence Model, Software Architecture

1. PRINCIPLES

OMax [2][4][6] is a software environment oriented towards human-machine interaction for musical improvisation. It learns in real-time by listening to an acoustic musician and extracting symbolic units from this stream. It then builds a sequence model on these units constituting an internal knowledge. The internal model of OMax (named Factor Oracle [1][5]) is a graph which incrementally recognizes the repeated factors (patterns and subpatterns) of any symbolic string. Factor Oracle only needs strict on the symbolic units to be built. They can be called *letters* over a formal *alphabet*.

OMax is able to navigate through this model to create one or several "clones" of the musician feeding the system [2]. These "clones" are recombinations of the original discourse justified by the model and realized by cutting and pasting the original material in real-time (audio editing or MIDI deslicing, see 2.1 and [4]).This *stylistic reinjection* [4] creates a specific musical interaction in which the musician is constantly confronted to a reinterpreted version of his own playing. It emphasize the memory effects and usage of (self-)reference found in improvisation contexts such as collective free improvisation or jazz.

NIME'12, May 21 – 23, 2012, University of Michigan, Ann Arbor.

This technique, close to concatenative synthesis, has been extended in a video version of OMax [6]. The video of the musician (or any other visual material) is aligned and recombined as a slave synchronization with audio.

Previous versions of OMax made use of two very different software environment, Max/MSP and OpenMusic (hence its name "OMax") respectively for real-time signal processing and abstract model construction and navigation processes. In this paper we will present a new version of OMax we developed solely based on Max5. We will describe the material it is now able to "understand" and how it achieves this. Then we will explain the architecture of this new version and its novel visualization and interactions possibilities. Finally we will discuss a few situations we encountered testing with musicians.

2. MATERIAL

Historically, OMax emerged from studies on *stylistic simulation* by Shlomo Dubnov and Gérard Assayag and Marc Chemillier's research on improvisation modeling. It has since gained considerable attention from improvising musicians worldwide through dozen of concerts, workshops and master-classes.

2.1 Audio vs. MIDI

Both audio and MIDI streams can constitute a source for OMax learning. Though MIDI is already a flow of abstract data, it still needs to be segmented into consistent units to be learnt. In the case of a monophonic MIDI input , segmentation is trivial: a unit for a note. However a polyphonic MIDI input feeds a continuous and overlapping flow of notes to be separated into polyphonic chord-like *slices* (Figure 1). This slicing happens with the birth and death of significant events and has been described in [4]. It does not imply any specific labeling (or lettering) to tag the symbolic units to be compared and learnt.

In the case of an audio input, prior to any kind of grouping, information needs to be extracted from samples. We have in OMax two very different types of audio analysis which infer two different kind of *listening*. The first type of analysis is pitch extraction. For now, we are able to deal only with monophonic pitch extraction and use the YIN algorithm [8]. To make the output of yin more consistent and stable, we use a statistical analysis with concurrent voting agents gathering pitches over fixed windows [3]. Stable pitches are gathered and grouped into units when equal and consecutive to form notes. At this point, we are brought back to the simpler case of monophonic MIDI-like data.

We summarize the different steps to form consistent units for the different type of analysis in Figure 1. From an *audio* stream, *micro-units* are constituted with an initial *framing* and the *extraction* of a descriptor of the signal. Depending

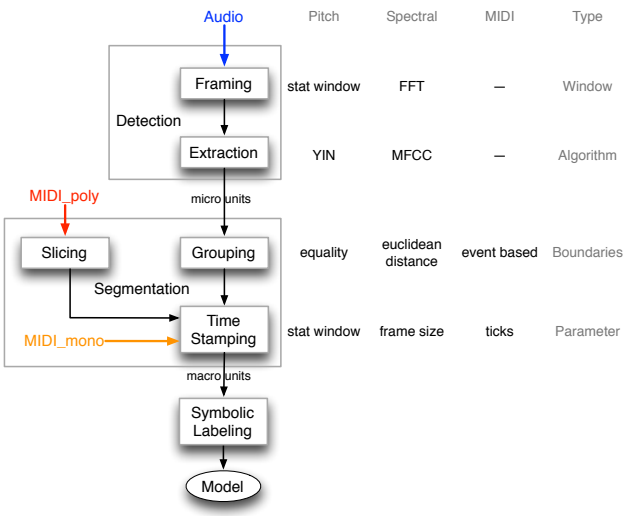


Figure 1: Analysis chain of the input

on the *algorithm* of extraction, different sliding *windows* are used. Then, in a second phase, *macro-units* can be put together by *grouping* similar and consecutive micro-units. The *boundaries* are defined by different criteria depending on the type of units. Then, to achieve the *segmentation*, we *time-stamp* these units linking them with the chunk of signal they encode — and we take into account the latency and time specific *parameters* of each methods.

2.2 Spectral Segmentation

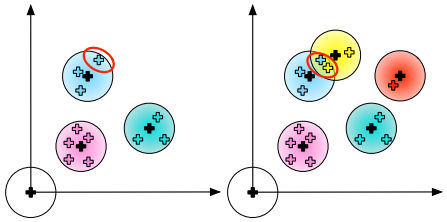
The second type of audio analysis in OMax uses spectral descriptors. They allow the system to *listen* and play with wider types of sound including noises, percussions or different instrumental playing modes. Mel Frequency Cepstral Coefficients has been proven (for example in [7]) to be easily computable, very compact and reliable for recognition tasks. It is suitable to constitute our micro-units. However, MFCCs are vectors of floating-point multi-scale coefficients. It is thus necessary to have a clustering algorithm to put together consistent and meaningful macro-units.

Rather than using a rough quantization as in [6], we designed a weighted Euclidean clustering algorithm. This algorithm achieves both the macro-grouping and the symbolic labeling at once.

For every incoming MFCC vector, dropping the first coefficient (which represent the overall energy of the slice), we weight the remaining coefficients according to profiles to help enhancing the differences we want to discriminate. These profiles have been adjusted empirically along experiments with several instruments and playing styles. Then we compare the vector to the clusters already encountered by computing the Euclidean distance with their centroids and we determine (with an adjustable threshold) if it can be identified or if it constitute a new letter in our spectral alphabet.

2.3 Alphabets

This incremental clustering mechanism creates a growing alphabet of spectral letters in a multidimensional Euclidean space, meaning that the system is able to discover the symbolic units along the musicians playing. It allows us to have an ad-hoc definition of the clusters depending on the input material. Regions of the timbre space thoroughly explored by the musician will have therefore more clusters than other regions, less brought into play. On the other hand, pitch classes and quantized spectral vectors constitute a fixed and predetermined alphabet.



(a) Clusters (b) Clusters later on

Figure 2: Example of mutation in spectral clustering

Another effect of this classification is the possibility of mutations according to the past of the learning. Depending on the material already encountered and known in the system — which means in our MFCC space, depending on the clusters already defined —, the same (or close) spectral vectors can be identified differently. An example of such a mutation is given in 2D Figure 2: vectors framed in red, although very close, are recognize differently depending on the moment they appear. The first vector (Figure 2a) is considered in the "blue" class while the second occurrence is closer to a more recently defined "yellow" cluster (Figure 2b). The appearance of the "yellow" cluster in between did not change the class of the previously encountered vector but it modifies the classification of forthcoming material. This also has a musical meaning: a specific playing mode can be considered as an accident and identified as close to an already heard mode if encountered only once. But the same playing mode, if developed by the musician may be rightfully creating one or more new cluster(s) — letter(s) in our spectral alphabet —to describe its slighter differences. Thus the mapping between signal units and symbols has become adaptive instead of being rigid, which reflects an important aspect of implicit learning in human interaction.

3. ARCHITECTURE

Further than being able to use different segmentations and alphabets, the whole software has been reprogrammed to adopt a very flexible and versatile architecture presented Figure 3.

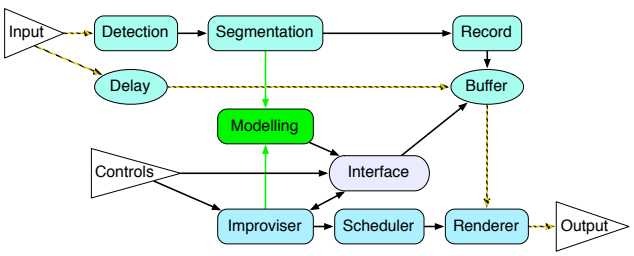


Figure 3: Functional diagram of OMax 4.x

3.1 Modularity

First of all, a modular approach has been taken to develop the different functions needed (Figure 3). The audio input is split in two streams. One is directly *recorded* into a *buffer* while the other enters a two stage process to constitute macro-units. The *detection* stage regroups both the *framing* of the signal and the *extraction* of a descriptor to get micro-units. Then the *segmentation* stage is in charge of the *grouping* and the *time-stamping* to define macro-units

and date them (see 2.1 and Figure 1).
Thanks to a fixed *delay*, the analysis chain described in the previous paragraph has a retro-action on the recording to start and stop it consistently (mainly avoiding to record long silences).

Once labelled (see 2.3), the symbolic units are fed incrementally to the model which will be read and navigated by improvising agents. To create a new "clone", the *improviser* is in charge of reading and jumping in the graph to create a new coherent path — a musical variation on what the acoustic musician played until now. The *scheduler* reading this path puts it back "in time" with the possibility of time-stretching or backward reading. Finally, the *renderer* effectively reads and crossfades the different *buffer* chunks corresponding to this new musical discourse.

3.2 Parallelism

The modularity of the new design allows now OMax to run in parallel different analysis and labeling and to acquire this way a multi-description model on a single input. The most typical setup making use of this is to run both the pitch and spectral analysis on a single audio input, building thus two Factor Oracles which refer to the same buffer and time-stamping.
Another very common option the new architecture allows, is to have several independent "clones" improvising on the same model. For that, we duplicate the whole generation chain, *improviser-scheduler-renderer*. Each *improviser* is able to have its own path on the common model of the input.

Besides these simple configurations, more sophisticated schemes are possible to create different musical linkage inside OMax. An example of these configurations is to have two "clones" playing the same computer-based improvisation — possibly at different speed or with transposition — ie. the same path in the model.

3.3 Versatility

Multiplying and combining at will the modules — the only limit being the power of the computer — we can shape and adapt our OMax setup to very diversified musical situations. From one to several inputs in parallels with one or more descriptions and models built on each of them and one to several "clones" improvising together or independently, the variety of arrangement allows us to start playing OMax almost as an instrument. We will see in 5 how OMax can now take its own musical part in different musical ensemble.

4. VISUALIZATION

On top of the redesign of the architecture, OMax 4.x adds to the software a whole new interface (thanks to Jitter, the graphical part of Max5). This interface is based on a visualization of the current state of one or two models being built. It takes the simple form of a growing horizontal and linear timeline representing what has already been learnt — time "flows" from left to right: left is farther in the past, right is the most recent element. Above and below this timeline can be shown some links of the Factor Oracle graph indicating the occurrences of repeated patterns. An example is presented Figure 4.

4.1 Live

Although unadorned, this feedback constantly given on the current state of the model revealed itself to be very efficient to locate and memorize on the fly musically interesting points. Seeing patterns of links appearing on the screen related to what the musician is currently playing allows to associate musical passages with particular sections of the graph. And retrieve them easily later on.

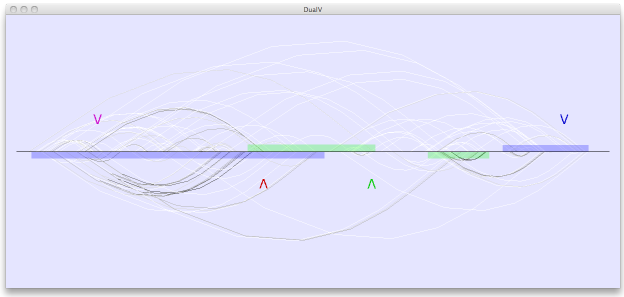


Figure 4: "Clones" and regions on the sequence visualization

While playing with OMax, the visualization is also a new interface to interact with the software. Indeed, as introduced Figure 4, the different "clones" OMax is able to play are also pictured on the display with moving arrows above and below the timeline. These arrows reflect the current position of each "clone" in the model and jump along the links when improvising new paths (see [2]). With the mouse, regions of the graph can be selected (green and blue sections on Figure 4) to constrain "clones" to specific sections of what has been learnt.

Many other features and functions that can not be detailed here have also been implemented to make the visualization as intuitive, interactive and useful as possible.

4.2 A Posteriori

Following the same visualization principles with different musical materials and visuals, we noticed that examining our model, a posteriori, could reveal important higher level structures. Moreover, this analysis may show, in real-time, these higher level formal structures with little graphical processing. For example, with colors to identify patterns and thickness to stress the durations, we can enhance certain aspects of the analysis and help discover interrelations of parts.

Figure 5 shows the analysis of a saxophone improvisation.

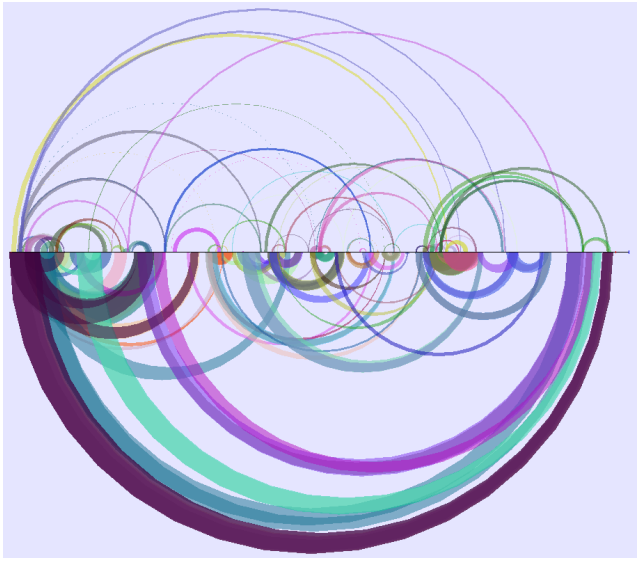


Figure 5: Visualization with colors and thickness

On the bottom half (below the horizontal timeline, which correspond to the model built on pitch), we can see with

the important number of arches, that the material introduced at the beginning of the session is developed for about a third of the total duration then the rest of the improvisation is very different (few links/arches connecting it with the beginning). Finally a short but strong recapitulation of the first material is played: many arches connecting the last moments of the improvisation with the beginning.

This way, the new interface of OMax may open possibilities for a simple yet efficient musicology tool to study pieces or improvisations and compare them.Automatic segmentation of formal high level structures in musical sequences has been experimented as a derivation from this visualization but is not presented in this paper.

5. OMAX IN ACTION

In the last two years, we had the opportunity to play with OMax in numerous and various situation. Here are some of the rough observations we could empirically make. We encountered mainly two kinds of musical ensembles. The first is a duet between an *acoustic musician* improvising and feeding the system and an *electronic musician* controlling the software. The second common situation is to include OMax (and the person controlling it) into a small musical group (usually up to ten musicians).

5.1 Duets

Naturally, the first idea to try out a software like OMax is to ask a musician to feed the system with an improvisation and play along with the computer based "clones". This allows us to test at the same time how the system *understands* the musicians playing and how they musically interact. Multiplying this kind of experiments with very different instruments helped us refine the different analysis stages of the software. While multiplying the different styles showed us the musical possibilities and limits of such a dialog.

It soon appeared in this situation that the choices made by the person controlling OMax strongly influence the course of the session. Thus, the acoustic musician could notice very fast the interventions and the character of the electronic musician, and often referred to this partner as a mixed entity constituted of OMax and the person controlling it, rather than as the software by itself. Duets of short duration work very well most of the time and some characteristics of purely acoustic improvised duets can be found.
However, when musicians want to make it longer (and confront it with a public), they usually feel more confident (pre)setting narrative elements. They frequently start to predefine the type of interaction with the system, mostly in terms of when to play and what part of the learning to use (which could be identified to a music score for OMax player). Or they feed OMax with some prepared material (pre-composed and/or pre-recorded) and improvise with these. Some previous experiments have been done in this last direction such as in *Peking Duck Soup* (Bloch 2008 [6]).

5.2 Groups

The other kind of musical situation OMax has regularly been involved in could be qualified as "collective free improvisation" or "band", that is a small group of musicians playing together, either improvising freely or with strong predetermined structuresIn this cases, one or two instances of OMax were used and each of them could *listen* and learn from one or two of the acoustic musicians. Technically, the question of feedback of other musicians and OMax into the microphones gets really serious in these situations.

Despite the difference of nature between the acoustic instruments and OMax — which usually does not have its own sound source — the computer and its player could really find a place comparable to the other musicians in the group. And the work to build a whole concert program was in almost every aspects similar to the work with an ensemble of only acoustic instruments.

6. FURTHER DIRECTIONS

Both the entire renovation and the musical testing feed the research around the musical aptness of OMax.

The explorations already started around the notion of alphabet in OMax constitute a strong direction of our future work. It puts up questions about the consistency of these alphabets and ways to build finer and even more meaningful alphabets. On-going research about information geometry or other content-oriented machine learning techniques may nourish our reflexion.
Multiplying the alphabets and models built in parallel gives opportunities to exploit more or differently the knowledge of the system. The various descriptions of an input sequence may be queried concurrently or jointly to help drive the system towards more expressive and speaking musical paths. This database oriented approach could use some reinforcement learning technique in order to make interesting solutions emerge.

Finally, numerous musical situation suggested we should investigate the relation of OMax with the rhythm both in a broad (phrase leading, endings...) and in a strict (pulse, time signature...) sense. A strong anticipation mechanism may be a relevant approach to these problems.

7. REFERENCES

[1] Allauzen, C., Crochemore, M., and Raffinot, M. Factor oracle: a new structure for pattern matching. In *SOFSEM'99* (Milovy, Czech Republic, Nov. 1999), vol. 1725 of *LNCS*, Springer-Verlag, pp. 291–306.

[2] Assayag, G., and Bloch, G. Navigating the oracle: a heuristic approach. In *International Computer Music Conference '07* (Copenhagen, Denmark, Août 2007), pp. 405–412.

[3] Assayag, G., Bloch, G., and Chemillier, M. Omax-ofon. In *Sound and Music Computing (SMC) 2006* (Marseille, France, May 2006).

[4] Assayag, G., Bloch, G., Chemillier, M., Cont, A., and Dubnov, S. Omax brothers : a dynamic topology of agents for improvization learning. In *Workshop on Audio and Music Computing for Multimedia, ACM Multimedia 2006* (Santa Barbara, USA, Octobre 2006).

[5] Assayag, G., and Dubnov, S. Using factor oracles for machine improvisation. *Soft Computing 8-9* (2004), 604–610.

[6] Bloch, G., Dubnov, S., and Assayag, G. Introducing video features and spectral descriptors in the omax improvisation system. In *International Computer Music Conference '08* (2008).

[7] Brown, J. C., Houix, O., and McAdams, S. Feature dependence in the automatic identification of musical woodwind instruments. *J Acoust Soc Am 109*, 3 (Mar 2001), 1064–72.

[8] De Cheveigné, A., and Kawahara, H. Yin, a fundamental frequency estimator for speech and music. *JASA: Journal of the Acoustical Society of America 111* (2002), 1917–1930.