# Fine-tuned Control of Concatenative Synthesis with CataRT Using the bach Library for Max

Aaron Einbond, Christopher Trapani, Andrea Agostini, Daniele Ghisi, Diemo Schwarz

# Fine-tuned Control of Concatenative Synthesis with CataRT Using the bach Library for Max

Aaron Einbond
HUSEAC
Harvard University

Christopher Trapani
CMC
Columbia University

Andrea Agostini
HES-SO
Geneva

Daniele Ghisi
HES-SO
Geneva

Diemo Schwarz
IRCAM–CNRS–UPMC
Paris

## ABSTRACT

The electronic musician's toolkit is increasingly characterized by fluidity between software, techniques, and genres. By combining two of the most exciting recent packages for Max, CataRT corpus-based concatenative synthesis (CBCS) and bach: automated composer's helper, we propose a rich tool for real-time creation, storage, editing, re-synthesis, and transcription of concatenative sound. The modular structures of both packages can be advantageously recombined to exploit the best of their real-time and computer-assisted composition (CAC) capabilities.

After loading a sample corpus in CataRT, each grain, or unit, played from CataRT is stored as a notehead in the *bach.roll* object along with its descriptor data and granular synthesis parameters including envelope and spatialization. The data is attached to the note itself (pitch, velocity, duration) or stored in user-defined slots than can be adjusted by hand or batch-edited using lambda-loops. Once stored, the contents of *bach.roll* can be dynamically edited and auditioned using CataRT for playback. The results can be output as a sequence for synthesis, or used for CAC score-generation through a process termed Corpus-Based Transcription: rhythms are output with *bach.quantize* and further edited in *bach.roll* before export as a MusicXML file to a notation program to produce a performer-readable score. Together these techniques look toward a concatenative DAW with promising capabilities for composers, improvisers, installation artists, and performers.

## 1. INTRODUCTION

Corpus-based concatenative synthesis methods (CBCS) [1] are more and more often used in various contexts of music composition, live performance, audio-visual sound design, and installation. They take advantage of the rich and ever larger audio databases increasingly available today to assemble sounds by interactive real-time or off-line content-based selection and concatenation. Fixed recordings or live-recorded audio are used to constitute the corpus, which makes the richness and fine details of the original sounds available for musical expression.

In parallel, composers have increasingly made use of computer sketching tools not only to produce work with an electronic component, but also work that will take an instrumental form. This includes programs designed specifically for the purpose of computer-assisted composition (CAC) like OpenMusic [2], PWGL [3], or the bach package for Max [4, 5]. It also includes programs designed primarily for other uses, like Digital Audio Workstations (DAWs), repurposed by composers to collage and model instrumental samples with an acoustic score-based goal.

However, tools taking advantage of traditional music notation, with possibilities for fine-tuned detail and editing, have generally remained separate from tools that offer a comparable level of control for audio synthesis. By combining concatenative synthesis with a flexible CAC package like bach, we can propose an advanced sketching tool for composers of both instrumental and electronic music. Corpus-based concatenative synthesis techniques like CataRT rely on audio features (or *descriptors*) to control synthesis at a high level [6]. These same audio features lend themselves to detailed music notation and CAC possibilities. CataRT's descriptor data can interact with bach's notation-based interface to benefit from the best of both worlds. Like a high-powered sampler, all of the synthesis parameters can be stored and edited dynamically. However unlike a traditional sampler, CataRT offers vastly richer possibilities for the organization of a dense sample micro-montage.

## 2. PREVIOUS AND RELATED WORK

Some efforts have already been made for fine control of granular micro-montage composition and concatenative synthesis in a sequencer-like environment.

Carlos Caires presented the granular synthesizer Irin in Max with sequencing, spatialization, and micro-editing of the generated random sequences, at each of three levels (micro, meso, macro) [7], constructed under the supervision of composer Horacio Vaggione.

Diemo Schwarz added MIDI output of generated grain sequences and playback capabilities to CataRT for the composition of the installation soundtrack *Trowel and Seal* (2008) in an external sequencer. Grains were represented as MIDI notes, their velocities would modify gain, and all transformation parameters were coded as control changes. However the limitations of the MIDI standard led to abandoning this approach. For instance, the grains' UnitID representation as notes stipulates distributing them over the 16 MIDI channels, but the resulting maximum number of grains ($128 \cdot 16 = 2048$) is still too small for working with

large corpora. Further, the control change representation of transformation parameters means they are dissociated from the grains they act upon, as well as lose precision, being mapped and quantized to 7 bit integers.

Instead the present implementation relies on two of the most exciting packages recently developed for MAX: CATART and BACH.

### 2.1 CATART

Corpus-based concatenative synthesis (CBCS) is based on segmentation and description of the timbral characteristics of the sounds in a database or *corpus*, and synthesis by selection of sound segments from the database matching sound characteristics given by the user. It allows one to explore a corpus of sounds interactively or by composing paths in the descriptor space, and to recreate novel timbral evolutions. CBCS can also be seen as a content-based extension of granular synthesis, providing direct access to specific sound characteristics.

It has been implemented in various systems and environments (see overviews of past [8] and present [1] approaches to CBCS) and notably in the interactive sound synthesis system CATART [6].

The CATART software system for MAX realizes corpus-based concatenative synthesis in real-time. It is a modular system based on the freely available FTM&CO extensions [2] [9], providing optimized data structures and operators in a real-time object system. CATART is released as free open source software. [3] There is also a standalone application version of CATART available, [4] and a new version based on the MUBU library [5] for MAX [10].

In addition to its capabilities for real-time synthesis, CATART has been used effectively for real- and deferred-time audio mosaicing and computer-assisted composition [11]. In both cases, a live or recorded audio input target is analyzed and compared to a preloaded corpus according to descriptors chosen and weighted by the user. This process may be termed "Corpus-Based Transcription" and the goal is to create a mosaic of samples form the corpus that best approximates one or more audio features of the target.

### 2.2 BACH

BACH is a library for MAX intended to bring CAC into the real-time world [4, 5]. The basic idea behind BACH is that symbolic score generation and modification is not necessarily an out-of-time activity: it can follow the composer's discovery process in real-time and adapt accordingly. BACH's hierarchic representation of data is directly inspired by the most common Lisp-based CAC environments such as OPENMUSIC [2] or PWGL [3]. BACH's nested lists, where hierarchies are defined via levels of parentheses, are indeed called *llll*s, an acronym for Lisp-like linked lists; see Figure 1 for an example. At the same time, BACH's approach to CAC takes direct inspiration from a digital signal processing model and applies such ideas to notes, chords, scores, and symbols in general.

At the center of the BACH environment are two score editors, *bach.roll* and *bach.score*, which provide flexible interfaces for the representation and modification of musical content. *bach.roll* expresses time in terms of absolute temporal units (namely milliseconds), while *bach.score* expresses time in terms of traditional musical units, and includes notions such as rests, measures, time signature and tempo. Each notation editor is equipped with advanced representation features such as support for microtonal accidentals of arbitrary resolution, variable play rate, and the possibility to associate custom meta-data to notes (see section 3.3). Moreover, editing can be performed both through the interface and with messages, which makes *bach.roll* and *bach.score* suitable for advanced real-time score handling.

## 3. ADVANCED RECOMBINATION OF MODULES

In order efficiently to implement CBCS with CATART and BACH, high-level capabilities of each system must be leveraged for the best performance.

### 3.1 Real-time BACH Usage

BACH is distinguished from other CAC tools by its real-time capabilities, therefore efficiency and cost are primary concerns. However as *bach.roll* and *bach.score* also support the flexibility of a user-friendly interface, they are not the most efficient by default. When real-time performance is a priority, several parameters can be disabled for optimization—undo, redraw during playback, and display of mouse-over legends—by setting the respective parameters maxundosteps, highlightplay, and legend to 0.

### 3.2 Lambda Loops

In computer programming it is often useful to combine two functions, operators, or objects in order to represent a specific sub-class of a wider process. Different programming languages offer different constructs to do this, and Lisp's *lambda functions* are probably among the most specific, powerful, and elegant. They are particularly relevant in the field of CAC because of the use of Lisp-based programming systems. MAX does not explicitly include such a concept; however its underlying callback-based structure easily allows the addition of a similar functionality to externals as well as abstractions. Several BACH modules, especially those implementing iteration-based operations, return single elements (*proposals*) from their rightmost outlet or outlets and expect an immediate return value for that proposal in their rightmost inlet. For example, an object filtering out elements from a list will propose elements one by one, and for each of them expect a return value expressing acceptance or refusal; an object sorting a list will propose pairs of elements, and for each pair expect a return value stating whether the pair is in the desired order or not. Return values are not always booleans: for example, the *bach.mapelem* abstraction proposes list elements

[1] http://imtr.ircam.fr/imtr/Corpus-Based_Sound_Synthesis_Survey
[2] http://ftm.ircam.fr
[3] http://ismm.ircam.fr/catart
[4] http://forumnet.ircam.fr/product/catart-standalone
[5] http://ismm.ircam.fr/mubu/

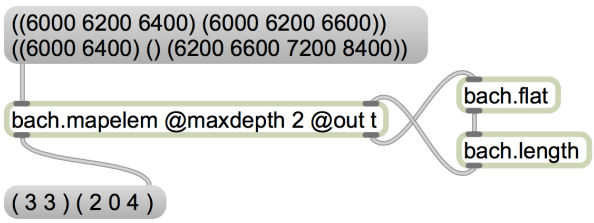


**Figure 1**: A simple example of a lambda loop: *bach.mapelem* iterates the incoming list up to two levels of depth, outputting elements from the right outlet one by one, and accepting the modified elements in the right inlet. The lambda loop (formed by *bach.flat* and *bach.length*) substitutes, for each sublist found at depth 2, its length.

to be substituted by their respective return values, mimicking Common Lisp's mapcar function (see Figure 1).

The name *lambda loop* has been chosen to identify this kind of conventional construct, as it is meant to provide the BACH user with a behavior replacing Lisp's lambda functions at least for simple cases. The portion of the patch providing this behavior begins and ends on the same object, forming a graphical loop. The outlets and inlets meant to be connected in a lambda loop configuration are called respectively *lambda outlets* and *lambda inlets*. Finally, in a well-engineered object or abstraction lambda loops never cause stack overflows or infinite loops, as lambda inlets are always "cold," meaning in MAX terminology that data they receive do not trigger subsequent callbacks.

### 3.3 Slots

The BACH notation objects, including *bach.roll*, allow a set of meta-data to be associated with each note, organized in containers called *slots* (see Figure 2). Slots are typed, covering a broad range of data structures including numbers, text, lists, breakpoint functions, matrices, filter definitions, and much more. Each notation object can currently support up to 30 slots per note, addressed by a numeric index or a name. Slot types and names are global: the type and name of the $n$-th slot are the same for all the notes in the score. However, slot types and names are freely assignable, as are most of the parameters characterizing their behavior (such as ranges, domains, hotkeys, etc.). Slot data can be inspected and edited both graphically and through MAX messages. When a *bach.roll* object plays back the score it contains, it outputs together with each note's data (such as pitch, velocity, duration) all associated slot data; this means that *bach.roll* can be used as an extremely flexible sequencer capable of driving nearly any kind of device or process, including audio DSP processes.

## 4. IMPLEMENTATION

The outlines of our implementation are traced in Figure 3 and described below.

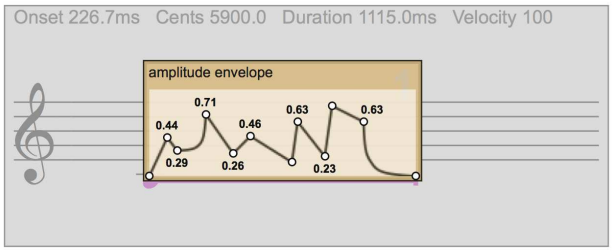


**Figure 2**: Each note in *bach.roll* stores pitch in MIDIcents, onset, duration, velocity, and a list of further synthesis data and metadata contained in slots. The image shows an open slot window, containing a breakpoint function.

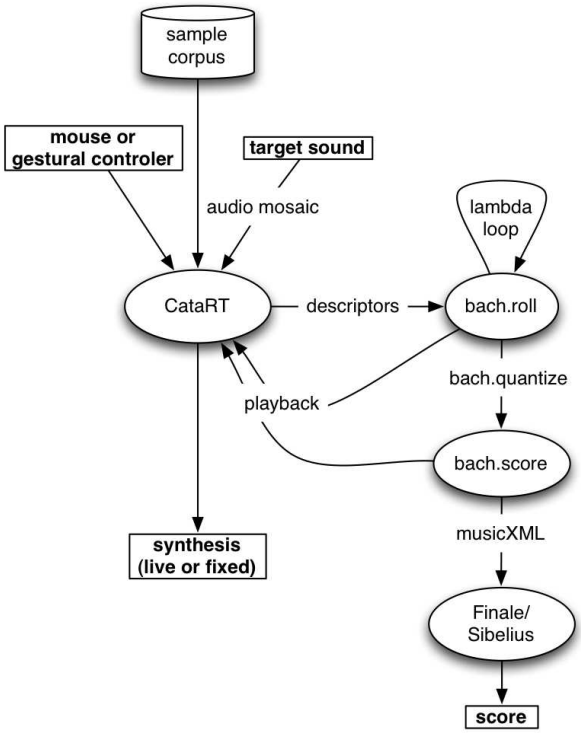


**Figure 3**: Flowchart for transcription, sequencing, and notation with CATART and BACH.

### 4.1 Recording CATART Synthesis with BACH

CATART is stocked with a corpus of audio, and synthesis is performed through gestural control of *catart.lcd* (with a mouse, tablet, or other controller), direct querying of *catart.select* according to a desired descriptor (like NoteNumber) or querying by a sequence of descriptors in time (as in an audio mosaicing task). Each CATART *grain* is stored in a *bach.roll* "score" as a notehead with its associated descriptors. The displayed pitch corresponds either to CATART's pitch estimate (NoteNumber) or to the pitch meta-data imported from sample filename (the user-defined descriptor FilenameNoteNumber).

Further synthesis parameters are recorded directly to *bach.roll*: onset, determined by the time elapsed since the start of transcription; velocity, converted from CATART's Loudness estimate; and duration, corresponding to the performed length of the grain (which may differ from the

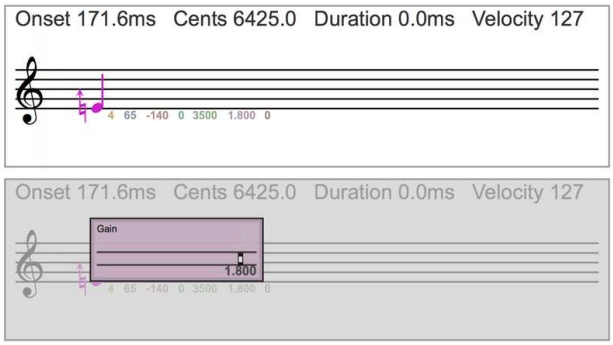


**Figure 4**: A CataRT unit is stored as a note in *bach.roll* including descriptor data contained in slots. In the first image slot contents are displayed as values attached to the note, while the second shows a slot window open for editing, in this case a floating point slider controlling gain.

length segmented by CataRT on import).

On playback, the score represented by BACH is played sequentially, and each note will trigger one grain in CataRT transformed according to the parameters represented by and attached to the note using BACH slots. Grains are spatialized according to parameters saved in slots as well.

### 4.2 Filling Slots

We use BACH slots to store the identification and playback parameters of each CataRT grain: most importantly UnitID, the grain's unique index in the corpus, which can be used by CataRT to look up any of the grain's stored descriptor data. Further granular synthesis parameters may also be stored in slots: as these are chosen dynamically by the CataRT user during synthesis, they cannot be otherwise retrieved from CataRT's stored descriptor values. These parameters include azimuth (in degrees), attack and release times (ms), gain (dB), and reverse flag (0 or 1).

While CataRT allows the user to select random granular synthesis parameters by specifying the random variation of grain length, transposition, gain, and panning, we prefer to set these values to 0 and instead control them directly outside of CataRT. This allows the data recorded from CataRT in *bach.roll* to be completely reproducible on playback (see Figure 4).

### 4.3 Live Editing of *bach.roll*

The *bach.roll* object offers a range of intuitive possibilities for editing CataRT's transcribed output. Notes can be deleted, dragged vertically for transposition or horizontally to change onset times. Durations can be lengthened or shortened by resizing the tails of each notehead. A selection of *bach.roll* can be shifted or stretched (in time or in pitch) by using dilation rectangles (see Figure 5). They are obtained by selecting a portion of *bach.roll* and pressing the Command (Macintosh) or Control (Windows) key before releasing the mouse button. All of these changes take effect on playback. Significantly, the stored and manually edited transposition and duration are independent of the separately stored NoteNumber and Duration of the CataRT units themselves, allowing for comparison of the desired values with the original sample.

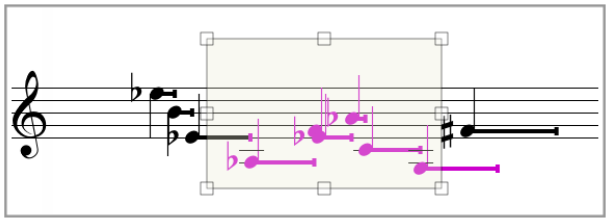

**Figure 5**: The dilation rectangle allows musical content to be stretched in pitch or time.

### 4.4 Batch Editing

Beyond pitches and onsets that can be graphically edited with the dilation rectangle, other granular synthesis parameters can be batch-edited using lambda loops. Any of the stored playback data or slot values may be altered by either a uniform addition or a percentage change. This is particularly useful to refine envelope shapes, gain, and spatialization parameters post-synthesis. [6]

### 4.5 Playback

Playback takes advantage of the native sequencing capabilities of the *bach.roll* object. As each note in the roll is triggered, its associated data is unpacked and sent to CataRT's synthesis module. The recorded playback parameters are reproduced before sending the UnitID as the final step, triggering the playback of the corresponding grain.

## 5. MUSICAL RESULTS

### 5.1 Generating an Orchestral Score

Using CataRT and BACH, a full orchestral score can now be automatically generated, subjectively edited, and efficiently exported to a music notation program for further alteration. To this end, a corpus of orchestral samples is imported into CataRT, which is then used to synthesize a concatenative montage. The montage could be produced using a gestural controller like a mouse or drawing tablet, or it could be the result of an audio mosaic where audio descriptors are compared to a target sound (Corpus-Based Transcription, see section 2.1), for example an ambient field recording. Whatever the source, the CataRT synthesis is recorded in *bach.roll* and saved for further manipulation.

#### *5.1.1 Targeted Transposition*

For contexts in which pitch-based sounds are favored, *targeted transposition* mode can be activated, in which CataRT transposes each unit to match a targeted NoteNumber. As mentioned above, *bach.roll* stores information to retrieve both the original pitch of the sample as well as the transposed pitch, so the user can make further adjustments.

[6] A demonstration of these editing possibilities can be viewed at https://vimeo.com/90281614.

#### 5.1.2 *bach.score and bach.quantize*

After subjectively editing the data in *bach.roll*, *bach.quantize* is used to transcribe the sequence into traditional rhythmic notation with *bach.score*. At each stage, the roll or score can be played back in real-time through CataRT, allowing immediate evaluation of the sonic results. Individual notes or units can also be played back one at a time (by highlighting the note and pressing "v" for *evaluate* and using the Command or Control key plus the arrow keys to navigate between adjacent notes) for detailed listening to individual samples. Before the quantization step, notes can be moved and distributed to different staves, for example to divide material between multiple instrumental parts. *bach.quantize* then presents several advantages over its predecessor in OpenMusic, including handling of grace notes, the possibility to define the minimal units of the quantization adaptively depending on the density of musical events, and accommodation of contrapuntal tempo changes including *accelerandi* and *decelerandi*.[7]

#### 5.1.3 MusicXML *Export*

From *bach.score*, the message exportxml exports the data to a MusicXML file readable by notation programs Finale and Sibelius. Upon import further subjective editing is necessary to produce a performance-ready score, for example by adding human-readable dynamics, performance technique text, and other indications not adequately exported through MusicXML. However the process is much accelerated compared to export with MIDI or by hand. This process was used to produce Aaron Einbond's *Endangered Sound*, five pieces for orchestra with field recordings (2013).

### 5.2 *bach.roll* as a Sequencer

Stored *bach.roll* objects also permit detailed control of synthesis, analogous to a highly flexible sequencer with fine control of playback and synchronization with real-time processes.

#### 5.2.1 *SoundSet Control*

CataRT is capable of dividing corpora into multiple SoundSets, groups of units that correspond to user-defined criteria such as instrument, source type, or source directory. The *soundset-control* module allows these groups of samples to be enabled or disabled, as well as for separate synthesis parameters to be applied to each SoundSet (see Figure 6). The possibility of altering envelope, gain, and spatialization parameters for subsets of the corpus, as well as for dynamic interpolations between these settings, allows for a much more detailed control over the synthesized result. SoundSets can be automatically segregated into separate staves in *bach.roll*, for a clear visualization of CataRT's output, and facilitating the independent editing of slots corresponding to each SoundSet.

[7] For a demonstration of this process from synthesis to quantization, please see https://vimeo.com/89840740.


**Figure 6**: A *soundset-control* module allows granular synthesis parameters for separate SoundSets to be edited and stored for playback.

#### 5.2.2 *Spatialization*

For spatialization purposes, we record for each unit an azimuth value, corresponding to the panning position, as well as a standard deviation value. This allows a stored *bach.roll* to be compatible with multiple spatialization modules and configurations. The multichannel synthesis module distributed with CataRT (*catart.synthesis.multi~*) receives for each unit a list of amplitudes for all output channels, as output by the spatializing tool *vbap*, and outputs to the Gabor overlap–add object (*gabor.ola~*) to control the spatial position of an arbitrary number of simultaneous grains. These tools were central to the composition of Christopher Trapani's *Writing Against Time* for two pianos, two percussionists and live electronics (2014).

With the change of a few arguments, the multichannel output can be reconfigured to accommodate a new setup, with the same *bach.roll* providing spatialization data. This adaptability is a major advantage over performing with multichannel soundfiles, avoiding the need to re-render in the new speaker configuration. Further spatialization parameters, for *vbap* or other tools, such as distance, spread, and elevation, could also be stored in slots.

## 6. CONCLUSIONS

### 6.1 Advantages of Real-time

Combining tools conceived for CAC and live concatenative synthesis in a single streamlined process, the distinction between real- and deferred-time applications begins to blur. Our results point toward a corpus-based concatenative DAW, allowing both high-level control of synthesis and a convenient user interface. The speed with which the user can audition, edit, and re-audition audio, from single grains to long sequences, offers dynamic feedback for a creative workflow where listening takes a central role.

In the context of fixed-electronic synthesis, bach storage presents several advantages over writing the results of CataRT synthesis directly to *sfrecord~*. The *bach.roll* object allows spatialization data to be easily written, stored,

and adapted to new speaker setups without the necessity to re-render soundfiles. Further, live concatenative synthesis allows for a flexibility in playback of a micro-montage of samples that cannot be achieved with bounced soundfiles. As the attack and decay of each unit in *bach.roll* is controlled separately, it is possible to interrupt playback in the middle of a sequence allowing the sounded units to decay naturally, an effect that cannot be easily simulated with a pre-mixed montage.

### 6.2 Further Directions

Several developments could improve the processes of transcription and synthesis further: first, more use could be made of the advanced capabilities of BACH slots. For example in the case of synthesis, rather than relying upon CataRT's attack and release parameters, a fully-editable break-point function (BPF) like that shown in Figure 2 could be used to shape the envelopes of grains for playback.

Or in the case of corpus-based transcription, better use could be made of the contents of slots upon export to MusicXML: meta-data associated with grains such as filename or directory could be included, giving access to important annotations like instrumental performance technique. This meta-data could then be mapped to noteheads, articulations, or other symbols, speeding the score editing process once the file reaches FINALE.

To expand sequencing possibilities, the playback of *bach.roll* objects could be coordinated with the MAX *transport* object to join with other time-based MAX objects in a consistent way. Further, our technique could be used with other existing score-based techniques such as score-following. In an initial test with *Antescofo* comparing a pre-loaded score to a live performance, detected notes were recorded in *bach.roll* along with tempo and certainty parameters stored as slot values. In addition to providing a recorded graphical verification of real-time score following performance, this also suggests more advanced recombination possibilities.

Finally, the powerful combination of CataRT and BACH could be integrated into a more extensive Feature Modulation Synthesis framework as proposed in [12], where not only pitch, dynamics, and spatialization, but other descriptors like SpectralCentroid could be adaptively altered to desired targets for re-synthesis.


### Acknowledgments

The work presented here was partially funded by the *Agence Nationale de la Recherche* within the project *Topophonie*, ANR-09-CORD-022.


## 7. REFERENCES


[1] D. Schwarz, "Corpus-based concatenative synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 92–104, 2007, special Section: Signal Processing for Sound Synthesis.

[2] G. Assayag et al., "Computer assisted composition at Ircam: From patchwork to OpenMusic," *Computer Music Journal*, no. 23 (3), pp. 59–72, 1999.

[3] M. Laurson, M. Kuuskankare, and V. Norilo, "An overview of PWGL, a visual programming environment for music," *Computer Music Journal*, vol. 33, no. 1, 2009.

[4] A. Agostini and D. Ghisi, "*bach*: an environment for computer-aided composition in Max," in *Proceedings of the International Computer Music Conference*, Ljubljana, Slovenia, 2012.

[5] ——, "Real-time computer-aided composition with *bach*," *Contemporary Music Review*, vol. 32, no. 1, pp. 41–48, 2013.

[6] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-Time Corpus-Based Concatenative Synthesis with CataRT," in *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, Montreal, Canada, 2006.

[7] C. Caires, "Irin: Micromontage in graphical sound editing and mixing tool," in *Proceedings of the International Computer Music Conference*, Miami, Florida, 2004.

[8] D. Schwarz, "Concatenative sound synthesis: The early years," *Journal of New Music Research*, vol. 35, no. 1, pp. 3–22, 2006.

[9] N. Schnell, R. Borghesi, D. Schwarz, F. Bevilacqua, and R. Müller, "FTM—Complex Data Structures for Max," in *Proceedings of the International Computer Music Conference*, Barcelona, Spain, 2005.

[10] N. Schnell, A. Röbel, D. Schwarz, G. Peeters, and R. Borghesi, "MuBu & friends—assembling tools for content based real-time interactive audio processing in Max/MSP," in *Proceedings of the International Computer Music Conference*, Montreal, Canada, 2009.

[11] A. Einbond, D. Schwarz, and J. Bresson, "Corpus-based transcription as an approach to the compositional control of timbre," in *Proceedings of the International Computer Music Conference*, Montreal, Canada, 2009.

[12] A. Einbond, C. Trapani, and D. Schwarz, "Precise pitch control in real time corpus-based concatenative synthesis," in *Proceedings of the International Computer Music Conference*, Ljubljana, Slovenia, 2012.