**Preprint version**. This is the author's final version, the article has been accepted for publication in The Computer Music Journal: Anna Xambó, Alexander Lerch, and Jason Freeman (2019). Music Information Retrieval in Live Coding: A Theoretical Framework. *Computer Music Journal*, 42:4, Winter 2018, 9–25. Please refer to the published version here: https://doi.org/10.1162/comj_a_00484.

# Music Information Retrieval in Live Coding: A Theoretical Framework

Anna Xambó

Department of Music

Norwegian University of Science and Technology

7491 Trondheim

Norway

anna.xambo@ntnu.no

Alexander Lerch

Center for Music Technology

Georgia Institute of Technology

840 McMillan St NW, 30332 Atlanta (GA)

USA

alexander.lerch@gatech.edu

Jason Freeman

Center for Music Technology

Georgia Institute of Technology

840 McMillan St NW, 30332 Atlanta (GA)

USA

jason.freeman@gatech.edu

## Abstract

Music information retrieval (MIR) has a great potential in musical live coding (LC) because it can help the musician/programmer to make musical decisions based on audio content analysis (ACA) and explore new sonorities by means of MIR techniques. The use of real-time MIR techniques can be computationally demanding and thus they have been rarely used in LC and with a focus on low-level feature extraction. This article surveys and discusses the potential of MIR applied to LC at a higher musical level. We propose a conceptual framework of three categories: (1) audio repurposing, (2) audio rewiring and (3) audio remixing. We explored the three categories in live performance through an application programming interface (API) library written in SuperCollider, MIRLC. We found that it is still a technical challenge to use high-level features in real time, yet using rhythmic and tonal properties (mid-level features) in combination with text-based information (e.g., tags) helps to achieve a closer perceptual level centered on pitch and rhythm when using MIR in LC. We discuss challenges and future directions of utilizing MIR approaches in the computer music field.


## Introduction

Music *live coding* (LC) is a music improvisation practice that is based on generating code in real time by either writing it directly or using interactive programming (Brown 2006; Collins, McLean, Rohrhuber, and Ward 2007; Freeman and Troyer 2011; Rohrhuber, de Campo, Wieser, van Kampen, Ho, and Hölzl 2007). *Music information retrieval* (MIR) is an interdisciplinary research field that targets, generally speaking, the analysis and retrieval of music-related information, with a focus on extracting information from audio recordings (Lerch 2012). Applying MIR techniques to live coding can contribute to the music generation process, both creatively and computationally. A potential scenario would be to create categorizations of audio streams and extract information on timbre

and performance content, as well as drive semi-automatic audio remixing, enabling the live coder to focus on high-level musical properties and decisions. Another potential scenario is to be able to model a high-level music space with certain algorithmic behaviors and allow the live coder to combine the semi-automatic retrieval of sounds from both crowdsourced and personal databases.

This article explores the challenges and opportunities that MIR techniques can offer to LC practices. Its main contributions are: **a survey review** of the state of the art of MIR in LC, **a categorization** of approaches to LC using MIR illustrated with examples, **an implementation** in SuperCollider of the three approaches, which are demonstrated, respectively, in test-bed performances and **a discussion** of future directions of real-time computer music generation based on MIR techniques.

# Background

In this section, we overview the state of the art of live coding environments and MIR related to live performance applications.

## Live Coding Programming Languages

The terms of *live coding language* and *live coding environment*, which support the activity of music live coding, will be used interchangeably in the following. Programming languages that have been used for LC include ChucK (Wang 2008), Extempore (previously Impromptu) (Sorensen 2018), FoxDot (Kirkbride 2016), Overtone (Aaron and Blackwell 2013), Sonic Pi (Aaron and Blackwell 2013), SuperCollider (McCartney 2002), TidalCycles (McLean and Wiggins 2010), Max/MSP and Pd (Puckette 2002), and Scratch (Resnick, Maloney, Monroy-Hernández, Rusk et al. 2009). With the advent and development of JavaScript and Web Audio, a new generation of numerous web-based LC programming languages has emerged, e.g., Gibber (Roberts and Kuchera-Morin 2012). Visual live coding

is also an emerging practice, notably with Hydra (github.com/ojack/hydra) and Cyril (cyrilcode.com).

### MIR in Commercial Software for Live Performance

Real-time MIR has been researched and implemented in mainstream software ranging from: **digital audio workstations (DAWs)** such as *Ableton Live*, **karaoke software** such as *Karaoki* (pcdj.com/karaoke-software/karaoki) and *My Voice Karaoke* (emediamusic.com/karaoke-software/my-voice-karaoke.html), **DJ software** such as *Traktor* (native-instruments.com/en/products/traktor), *Dex 3* (pcdj.com/dj-software/dex-3), and Algoriddim's *Djay Pro 2* (algoriddim.com) and **song retrieval or query-by-humming applications** such as *Midomi* (midomi.com) and *SoundHound* (soundhound.com). Typically, these solutions include specialized MIR tasks, such as pitch tracking for the karaoke software, and beat and key detection for the DJ software. In these applications, the MIR tasks are focused on high-level musical information (as opposed to low-level feature extraction) and, thus, inspires this research. There already exist collaborations or conversations between industry and research looking at real-time MIR applied to composition, editing, and performance (Bernardini, Serra, Leman, and Widmer 2007; Serra, Magas, Benetos, Chudy et al. 2013). To our knowledge, there is little research on the synergies between MIR in musical live coding and MIR in commercial software. The present article aims to help fill that gap.

## Real-Time Feature Extraction Tools for LC Environments

In this section, we present a largely chronological and functional analysis of existing MIR tools for LC and interactive programming to understand the characteristics and properties of the features supported, identify the core set of features, and discuss whether the existing tools satisfy the requirements of live coders.

*Table 1. Some MIR tools for LC environments.*

| Library Type | MIR Tool | Authors | LC Environment |
|---|---|---|---|
| *Included* | • *Unit Analyzer (UAna)* | Wang et al. (2007) | ChucK |
| | • *fiddle~*, *bonk~* and *sigmund~* objects | Puckette et al. (1998) | Max/MSP |
| | • *Built-in UGens* for machine listening | SuperCollider community | SuperCollider |
| *Third-party* | *SMIRK* | Fiebrink et al. (2008) | ChucK |
| | *LibXtract* | Bullock (2007) | Max/MSP, SuperCollider |
| | *Zsa.Descriptors* | Malt and Jourdan (2008) | Max/MSP |
| | *analyzer~* object | Tristan Jehan | Max/MSP |
| | *Mubu* | Schnell et al. (2009) | Max/MSP |
| | *SCMIR* | Collins (2011) | SuperCollider |
| | *Freesound quark* | Gerard Roma | SuperCollider |

Nearly all MIR systems extract a certain intermediate feature representation from the input audio and use this representation for inference, for example, classification or regression. Although still an open debate, there is some common ground in the MIR literature about the categorization of audio features between low-, mid-, and high-level. Lerch (2012) refers to *low-level features* as instantaneous features, extracted from small blocks of audio and describing various, mostly technical properties such as the spectral shape, intensity, or a simple characterization of pitched content. Serra, Magas, Benetos, Chudy et al. (2013) refer to tonal and temporal descriptors as *mid-level features*. *High-level features* usually describe semantic characteristics, e.g., emotion, mood, musical form, and style (Lerch 2012), or overall musical, cultural, and psychological characteristics, such as genre, harmony, mood, rhythm, and tonality (Serra, Magas, Benetos, Chudy et al. 2013). The distinction between low-, mid-, and high-level audio features is used here because it provides enough level of granularity that it can be useful to the live coder.

Various audio feature extraction tools for LC have existed for some time. In the following, we present an overview of existing tools for the three environments ChucK (Wang 2008), Max/MSP (Puckette 2002), and SuperCollider (McCartney 2002), because

of their popularity, extensive functionality, and maturity. The findings and conclusions, however, are generalizable to other LC environments. The selection of MIR tools for analysis, which aims to be illustrative, is described in Table 1.

Figure 1 shows a timeline of the analyzed MIR tools from 1990s to present, which provides a historical and technological perspective of the evolution of the programming environments and dependent libraries. As shown in Figure 2a, the LC environment that includes most audio features is SuperCollider, whilst the third-party libraries with the largest number of features are LibXtract and FFSound, the latter as a wrapper of the Essentia framework. Figure 2b shows the distribution of the audio features over 6 categories (statistical, envelope, intensity, timbre, temporal, and tonal), which are inspired by Essentia's organization (Bogdanov, Wack, Gómez, Gulati et al. 2013) and Lerch's feature classification (2012).

The timbre audio features are the greatest in number (43% of the total). The most popular spectral features are spectral centroid, spectral rolloff, spectral spread, and Mel Frequency Cepstral Coefficients (MFCCs), followed by the less popular but still prominent spectral flatness and spectral flux. Statistical audio features is the second category with stronger representation (16% of the total), with zero crossing rate as the most frequently implemented. The larger supported intensity audio features are loudness and RMS. In temporal audio features, beat tracker and onset detection are the most present. Finally, there is a considerable amount of tonal audio features (15% of the total), where pitch (F0) is the only feature supported extensively.

Considering the most popular features mentioned above, it is noticeable that the majority are low-level (instantaneous) features (e.g., RMS, spectral centroid). These features have been widely used in audio content analysis (ACA), for example, to classify audio signals. In LC, they can be helpful to determine audiovisual changes in real time, yet the amount of data can be overwhelming and most of the extracted features are not musically

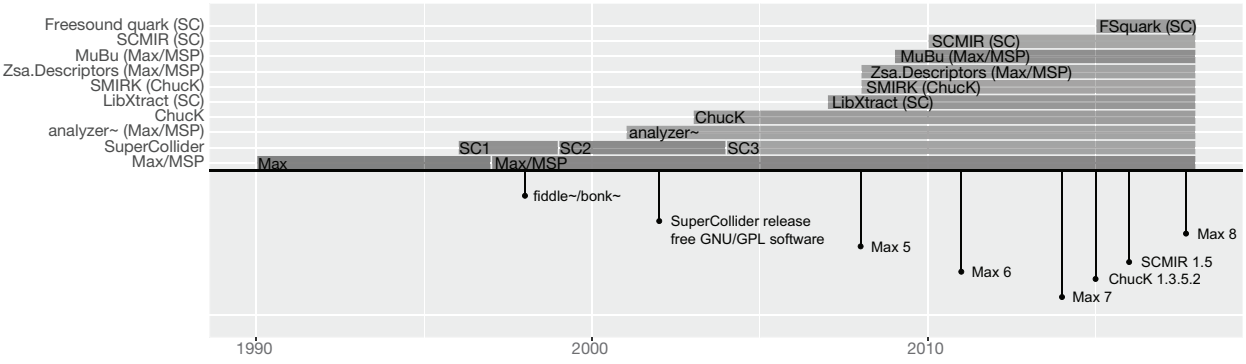
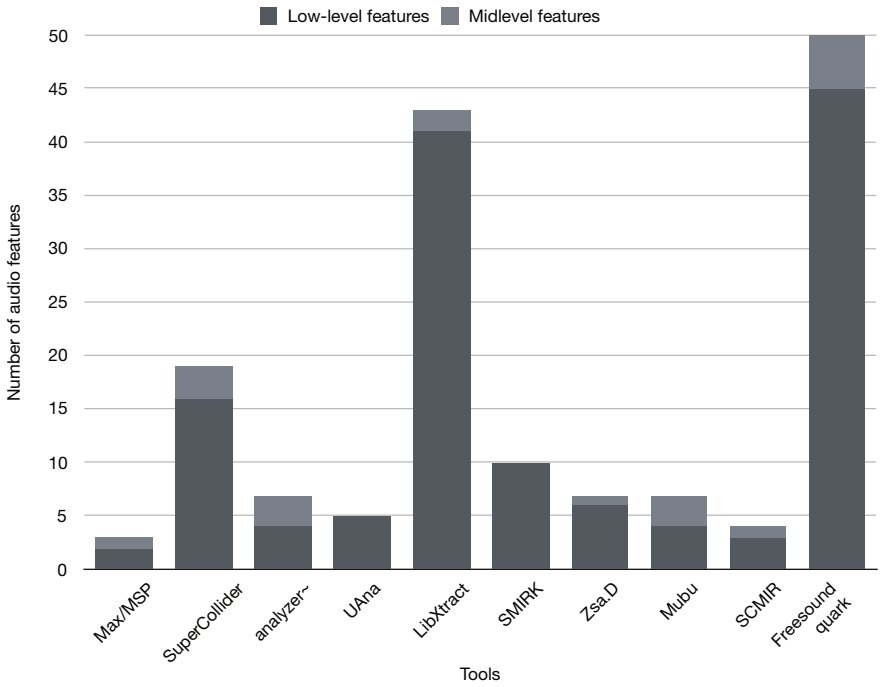


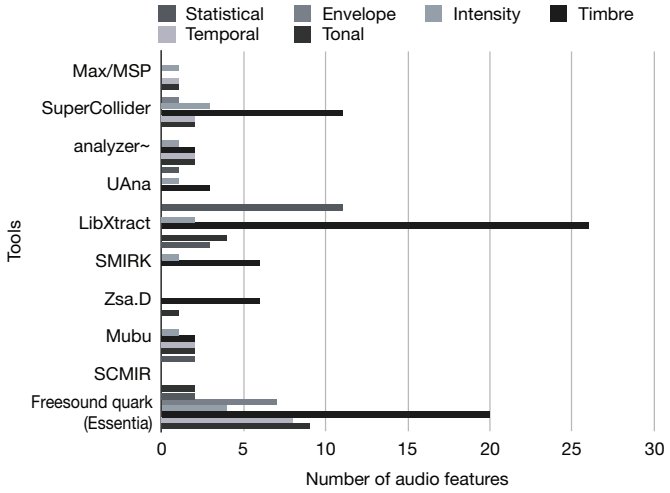

*Figure 1. Timeline of LC programming languages and dependent MIR tools.*

meaningful (e.g., how to interpret continuous float values of MFCCs). Mid-level features, such as onsets and tonality, can be easier interpreted and can be used to characterize rhythmic and tonal properties, supporting the live coder to make content-driven decisions (e.g., a threshold value can indicate whether or not to trigger a percussive sound depending on the onset values).

It is noteworthy that the presented real-time MIR tools provide the coder only with few mid-level features and show a lack of high-level features (Figure 2a). As later discussed in this article, the use of high-level features (e.g., mood, genre) requires a longer time scope, and systems extracting high-level features are often error prone (e.g., see the semantic gap problem between low-level descriptors and semantic descriptors (Schedl, Gómez, Urbano et al. 2014)). This may affect the performance negatively, compared to the immediacy and ease of use of instantaneous low-level features. At the same time, if the technical issues were solved, high-level information could increase the richness and musicality of the live coding session as the live coder could make decisions from a musical perspective of, e.g., the overall musical form.




*(a) Distribution of audio features by number and low-/mid-level audio features.*




*(b) Distribution of audio features by categories.*

*Figure 2. Functional analysis of MIR tools for LC.*

# Conceptual Framework on MIR in Live Coding

The presented conceptual framework structures MIR approaches for LC in three non-exclusive categories: (1) **audio repurposing**, by which we mean the analysis, retrieval, and manipulation of audio clips from a sound or music database, (2) **audio rewiring**, by which

we mean the real-time analysis of an audio input signal, which can be both an external source or the output of the system itself and (3) **audio remixing**, by which we mean a system supporting musically meaningful remixing decisions semi-autonomously.

For each of the three categories, we present three prototypes, respectively. The aim is to provide a conceptual foundation that helps us to discuss future possibilities. The three prototypes have been developed as three separate modules that comprise *Music Information Retrieval for Live Coding* (*MIRLC*) (github.com/axambo/MIRLC), an application programming interface (API) library written in SuperCollider that is designed to provide a musical approach to using MIR techniques in LC.

## Audio Repurposing

Since the advent of the social web, sound sharing and retrieval from online databases (e.g., Freesound, Looperman, ccMixter) has become increasingly popular (Font, Roma, and Serra 2017). Sounds are identified and retrieved by parsing either user-annotated data (e.g., tags, annotations, descriptions) or automatically extracted data. Font, Roma, and Serra (2017) identify two types of queries in audio retrieval based on features: (1) *queries by range* of a property, i.e., retrieval of sounds that match a given interval or threshold filtered by one or multiple audio features and (2) *queries by similarity*, i.e., retrieval of similar sounds from a given example. An example of the former is giving a range of bpm values to find specific rhythmic sounds. The latter has been notably explored for sound synthesis techniques such as musical mosaicing (Zils and Pachet 2001), concatenative sound synthesis (Schwarz 2007), or mashups (Serra, Magas, Benetos, Chudy et al. 2013) by retrieving short sounds similar to a reference. Other useful queries are by category, e.g., instrument or genre.

Figure 3 illustrates a block diagram of the audio repurposing approach in live coding, where the live coder can retrieve sounds from either online databases or local databases.

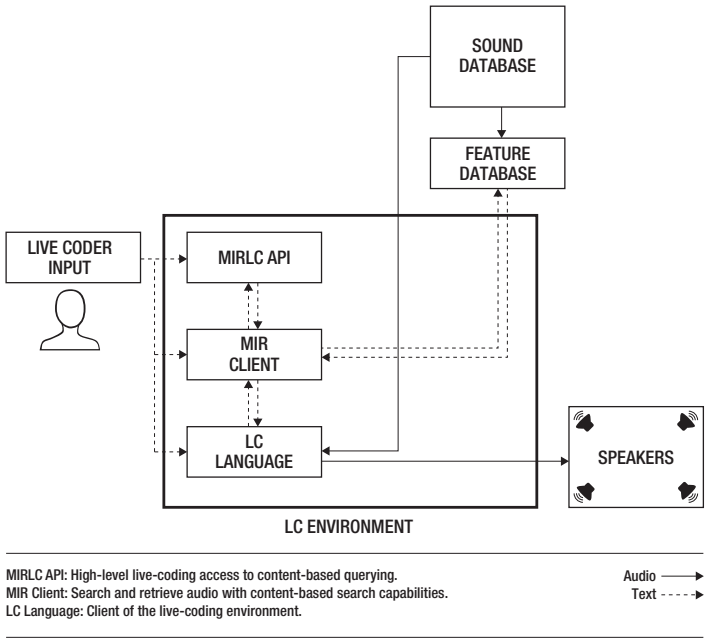



*Figure 3. Block diagram of audio repurposing.*

The retrieved sounds will be processed and integrated into the performance using live coding techniques. For efficient audio retrieval in quasi-real-time, a common practice is to pre-analyze the sound database and store its feature representation, thus combining the time-consuming offline analysis with ad-hoc access to pre-computed features and meta data of the audio clips (Serra, Magas, Benetos, Chudy et al. 2013). Typically, the offline analysis uses a feature extractor program and stores the information of each audio file as a single vector containing the aggregated audio features (Font, Roma, and Serra 2017). The LC language deals with low-level operations of real-time audio processing, such as loading sounds in buffers, which implies processing blocks of audio samples in real time.

*Examples of Audio Repurposing*

Current examples of audio repurposing in LC retrieve audio clips usually by textual queries from personal or online databases, as shown in LC coding sessions with Gibber (youtu.be/uly7DgtfRKI?t=96) or Tidal (youtu.be/FenTeBMkAsQ?t=275), as well as media clips with Live Coding YouTube (Lee, Bang, and Essl 2017), and even any type of input

data (Tsuchiya, Freeman, and Lerner 2016). Similarly, in Gibberwocky, multiple audio clips are also used from an already selected collection of sounds by means of a DAW interface such as Ableton Live connected to Gibber (Roberts and Wakefield 2017).

Exploring a wider range of scenarios, notable examples of machine listening systems can be found, particularly in art installations, live performances, and educational tools. These examples, which can inform future LC applications, include EarSketch (Xambó, Lerch, and Freeman 2016), Floop (Roma and Serra 2015), Freesound Explorer (Font, Roma, and Serra 2017), BBCut (Collins 2002), the LoopMashVST plugin (youtu.be/SuwVV9zBq5g), and APICultor (Ordiales and Bruno 2017), among others. These systems represent audio clips by content-based feature analysis, and often retrieve them in combination with text-based information (e.g., tags). With the exception of APICultor, they use a combination of low-level features (e.g., timbral properties) with mid-level features (e.g., temporal and tonal properties) for browsing sounds. Floop, for instance, looks into the beat spectrum to see how rhythmic a sound is. However, these systems are highly constrained to particular use cases. Most of them are based on audio clips constrained to particular features, e.g., their rhythm (Floop), beat (BBCut) or timbre (LoopMashVST). During a LC session, it can be desirable to retrieve sound samples without such constraints. According to Font, Roma, and Serra (2017), a sound database can include both short recordings (e.g., acoustic events, audio fragments) and long recordings (e.g., music, speech, environmental sound scenes). Low-level features, such as MFCCs, can be used in most types of sounds (e.g., environmental sounds, speech, music), while higher level features usually make assumptions of the input (e.g., detectable onsets and pitches, identifiable tonality, minimum length to detect rhythmic properties). As in APICultor, a LC system should have the flexibility of filtering the results by the choice of multiple audio features. However, there are computational challenges respective to feature aggregation that limit the length and content of this approach. Using an online database with pre-analyzed audio features seems like a workable approach for LC, which can be combined

*Figure 4. The first author livecoding with MIRLC at Noiselets 2017, Freedonia, Barcelona, Spain. Photo by Helena Coll.*

with a pre-analyzed local database, as discussed in previous work (Xambó, Roma, Lerch, Barthet, and Fazekas 2018).

The use of visualization tools (e.g., a two-dimensional space) for exploring the database content, as illustrated in EarSketch, Floop and Freesound Explorer, can allow for efficient browsing in a LC context. As discussed, this approach requires an offline analysis and each new query can disrupt the existing sound output if it is not designed for live performance. An example of an interactive visual timbral map for sound browsing suitable for live performance is found in a demo of SCMIR (youtu.be/jxo4StjV0Cg). As shown in this demo, using visualization techniques to show browse-and-query processes is a compelling addition to the LC practice, yet it broadens the definition of LC because visual media is added to the coding environment, as explored by others (Lee, Bang, and Essl 2017; McLean and Wiggins 2010; Roberts and Wakefield 2017; Tsuchiya, Freeman, and Lerner 2016).

*Prototype 1: A Case Study for Exploring Audio Repurposing*

This prototype aims at providing a high-level musical approach to operate with audio clips in LC using MIR, as previously discussed and assessed (Xambó, Roma, Lerch, Barthet, and Fazekas 2018). Figure 4 shows a test-bed performance using this module. It is designed for repurposing audio samples from Freesound using SuperCollider (online at vimeo.com/249968326). This module is built on top of the Freesound quark (github.com/g-roma/Freesound.sc), a SuperCollider client for accessing audio clips from Freesound through the Freesound API. The benefit of using the Freesound archive is that it allows to browse almost 400,000 sounds either by text search, content search, or a combination of both.

Inspired by the web interface Freesound Radio (Roma, Herrera, and Serra 2009), this module promotes loading sounds in a meaningful musical way. The live coder has a range of options to retrieve sounds, including mid- and high-level content-based queries (e.g., duration, bpm, pitch, key, scale) and text-based queries (i.e., tags). Sounds can be retrieved in groups, which facilitates the creation of conceptually-related sound groups based on similarity, rhythm, or pitch, among others. Each sound is played in loop and the groups can be played either simultaneously or sequentially. This provides different levels of musical granularity. The main functionalities include asynchronous management of multiple sounds by a single query or operation; human-like queries by content, similarity, tag, filter, sound id, and random; and an architecture to play with the groups of sounds either in sequence or in parallel. Figure 5 shows an example of the code.

In this case study, the role of LC is to query, browse, and control the audio clips in real time. LC sends out high-level textual (e.g., tags) and content-based queries (e.g., pitch, bpm, key, or scale), the latter based on MIR information, with the intent of crafting a coherent sound palette. Using MIR techniques removes the requirement of individually knowing each sound to create a homogeneous and coherent sound pool. The combination

```
// instantiation
a = MIRLCRep.new
b = MIRLCRep.new

// get sounds by content
a.content( 1, 'pitch', 22000, 'conf', 'hi' )
b.content( 1, 'dur', 0.01, 'conf', 'hi' )

( t = Routine( { // creation of a control structure for sequencing
    var delta;
    loop {
        delta = rrand( 2, 10 ); // generate a random number between 2 and 10
        if ( [ false, true ].choose, // choose with equal chance the value of false or true
                { a.similar }, // if true: get a similar sound from first sound in group a
                { b.similar } // if false: get a similar sound from first sound in group b
        );
        delta.yield; // amount of time in seconds until the routine should execute again
    }
} ); )

t.play // play the routine defined above

( r = Routine( { // creation of another control structure for sequencing
    var delta;
    loop {
         // generate a random number between 0.0005 and 0.3
        delta = rrand( 0.05, 3 ) * rrand( 0.01, 0.1 );
        if ( [ false, true ].choose,
                { b.sequence }, // if true: play sounds of group b in sequence
                { b.parallel } // if false: play sounds of group b in parallel
        );
     delta.yield;
    }
} ); )

r.play
```

*Figure 5. Example of code of the audio repurposing prototype (code extract from the album H2RI by the first author (pan y rosas, 2018)).*

of metadata with ACA provides flexibility and variation to the performance. It is possible to search for sounds based on different criteria such as rhythm, melody, duration and harmony. The use of the similarity descriptor can give musically consistent results yet they are still unpredictable, which evidences that defining music similarity is a nontrivial question (Lerch 2012). As future work, the MIR processes that are taking place could be made more visible to both the audience and the live coder using textual feedback, following the notion of 'showing the screen' in live coding. Next steps of interest include the use of even higher-level content-based features (e.g., mood, genre) to be combined with textual-based queries. In the current version, the use of tags has worked as a workaround

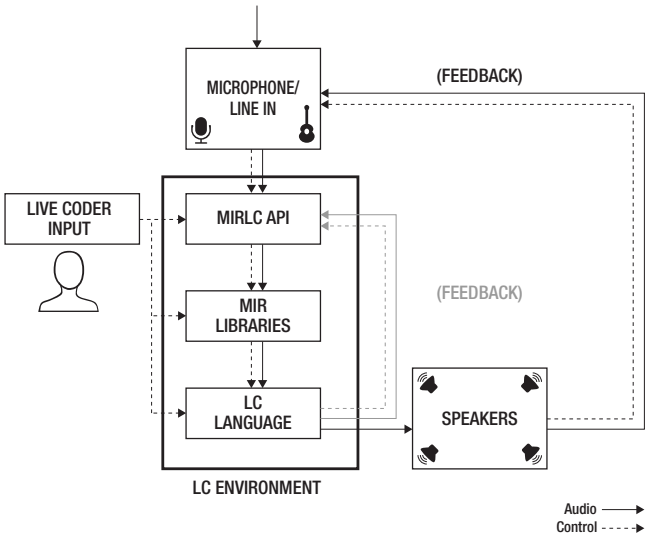



*Figure 6. Block diagram of audio rewiring.*

(e.g., "happy", "sad", "angry", "excited", "techno", "drumbeat", "dancehall").

## Audio Rewiring

The use of an audio stream as an input is a common practice in interactive computer music and machine listening (Chadabe 1984; Rowe 1993, 2001). The advent of sufficiently fast computers and suitable software has made feature analysis of the audio input signal possible in real time. Mapping the analysis results to sound processing parameters opens a range of creative possibilities for both studio and live performance, with prominent examples such as auto-tune or intelligent harmonizers. Furthermore, the audio output signal can be fed back to the input of the system to create a feedback system, a practice that dates back decades. Audio feedback has been extensively used in computer music, either as analog audio feedback, digital audio feedback, or both (Sanfilippo and Valle 2013).

Figure 6 shows the block diagram of the audio rewiring approach, in which the live coder receives an incoming audio stream (e.g., mic, line in, output of the system). The real-time analysis of this audio signal is used to define control or audio signals of the system. The LC language processes the incoming audio in real time using buffers.

*Examples of Audio Rewiring*

There are notable examples of live coding systems making use of this paradigm, including the BBCut2 library (Collins 2006), the Beatboxing classifier (Stowell and Plumbley 2010), the Algoravethmic remix system (Collins 2015), and McLean and Sicchio's (2013) Sound Choreography <> Body Code (vimeo.com/62323808). Beyond live coding, some examples combine hardware and real-time audio analysis, e.g., the Machine Listening Eurorack module (Latina 2016). Similarly, there exist sound installations with feedback and real-time analysis, such as Sanfilippo and Valle's (2013) Rumentario Autoedule (vimeo.com/37148011) and Agostino Di Scipio's Audible Ecosystems (Sanfilippo and Valle 2013). These examples extract numerous low-level features (e.g., spectral features) and a small subset of mid-level features (e.g., tonal and temporal characteristics). There exist, however, constraints that can limit their usability. In BBCut2, for example, the audio has to be recorded before being able to apply effects, which adds a delay not ideal in a real-time scenario. Similarly, in the Beatboxing classifier, even with workarounds in place, latency cannot easily be avoided and remains an issue for real-time performance. This indicates the importance of considering these constraints when designing an algorithmic or compositional system for performance. The feedback component of audio rewiring adds risk of failure to the performance and at the same time it can bring interesting results. Audio feedback can potentially make the overall system unstable, however, this instability can also be used artistically and can be creatively incorporated into a performance (compare, e.g., artists such as Sonic Arts Union, Loud Objects, Gordon Mumma).

*Prototype 2: A Case Study for Exploring Audio Rewiring*

This prototype aims at providing a high-level musical approach to operate with incoming audio (e.g., acoustic instrument, voice) in LC using MIR. It is designed for rewiring an audio input signal as either a control signal or audio signal using MIR techniques in SuperCollider (online at vimeo.com/249997271). An early version of the prototype has

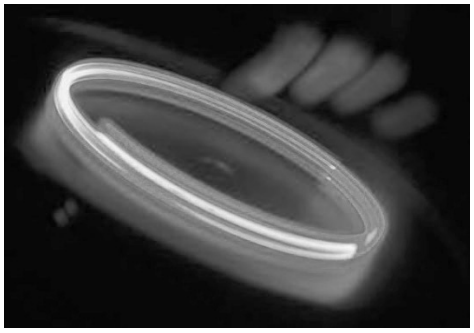


*Figure 7. Closeup of Anna Weisling's Distaff system, NIME 2017, Stengade, Copenhagen, Denmark. Photo by Jimmi Brandt Fotos.*

been used in the piece *Beacon* (Weisling and Xambó 2018), a collaborative audiovisual work between a visualist and an electronic musician, which has been internationally acclaimed (Lee, Jo, Weisling, Xambó, and McCarthy 2018). The visualist works with the Distaff system (see Figure 7), a DIY modified Technics turntable (Weisling 2017). The sound of the performer's fingers interacting with the turntable (e.g., scratching, tipping) as well as the inner mechanical sounds of the device produced from these interactions, are captured with a lavalier mic located inside the wooden box that contains the mechanical parts of the original turntable. Audio features extracted from the audio input signal either control effects applied to the audio signal or parameters of other audio signals. In this prototype, we explore unidirectional control (as opposed to a feedback loop).

An example of code is shown in Figure 8. The use of an audio input signal with different roles throughout the piece (e.g., audio signal, control signal) provides versatility and a wide range of variation. LC is used to change the role of the audio input signal. Feature analysis of the audio signal is applied to either control a sound generator (e.g., an estimated beat from the source system triggers a kick drum sound) or to modulate an effect parameter (e.g., an estimated onset from the source system modulates a multigrain effect). The rhythmical nature of the audio source shapes the mappings, thus the mapping design can be reconsidered when using a less percussive sound source, e.g., a voice.

```
// instantiation
a = MIRLCRew.new( 1, "afro-beat-6-8-toms.wav" )

// starting onset detection with a beep style sound
a.onsets( 'beep', 1 );

( r = Routine( { // creation of a control structure for sequencing
    var delta, option;
    loop {
        delta = 4.0;
        delta.yield;  // amount of time in seconds until the routine should execute again

        // if the modulus of 4 from the total number of onsets has remainder 0
        // (4 clockwise sequence)
        if( MIRLCRew.counter % 4 == 0,
                {
                // choose with equal chance one of the 4 options
                option = [ 'pitch', 'beats', 'onsets', 'amps' ].choose;
                case
                  {option == 'pitch'}
                        { a.pitch } // pitch follower
                  {option == 'beats'}
                        { a.beats } // beat tracker
                  {option == 'onsets'}
                        { a.onsets('beep', 1) } // onset detector
                  {option == 'amps'}
                        { a.amps('spark') } // peak amplitude tracking with a percussive sound
                });
    }
} ); )

r.play // play the routine defined above
```

*Figure 8. Example of code of the audio rewiring prototype.*

The presented prototype uncovered some conceptual drawbacks. Although a high level of synchronicity was noticed, the audience was unaware of the role of LC for generating sound and the general system setup involving the audio/control signal from the turntable. Furthermore, in collaborations between a live coder and a visualist, there can be a conflict of how to make both processes visible when visuals are so prominent. Previous research discusses that both approaches (opacity vs. transparency) are equally used and combined in audiovisual performance (Weisling, Xambó, Olowe, and Barthet 2018). The decision of creating a more immersive environment by not focusing the audience's attention on a screen with code extends the spectrum of live coding practices. Given the nature of the piece, a sound palette and algorithmic rules had to be defined ahead for each section, where improvisation was determined by the score of the piece. As future work, we

plan to design a less constrained environment by adding more flexibility to the mappings so that more ad hoc decisions can be made in real time, similar to UrSound (Essl 2010) and Gibber (Roberts, Wright, Kuchera-Morin, and Höllerer 2014). As an interesting future work, a challenge is the use of higher level audio features (e.g., mood, genre) to make decisions in real time. This would require a latency for accurate classification of events and the application of strategies for working around this processing delay, as discussed in Stowell and Plumbley (2010).

## Audio Remixing

There is a long tradition of network music in computer music (Weinberg 2005; Xambó 2015). The key terms and characteristics of network music include different types of network organization (e.g., centralized vs. decentralized, hierarchical vs. egalitarian), roles between performers (e.g., soloist vs. accompanist, producing sound vs. modifying sound), control (e.g., shared vs. individual), and types of contribution (e.g., parallel, serial, circular, multi-directional) (Xambó 2015). This is helpful to understand the nature of network music examples and how LC takes place using MIR on a musical network, typically applied to live remixing multiple audio streams.

Figure 9 outlines a block diagram of the audio remixing approach, where the live coder receives multiple audio streams (e.g., mic, line in, output of the system, and so on). The real-time analysis of the audio signals is used to help the live coder taking semi-autonomous decisions about the live remix.

*Examples of Audio Remixing*

There are illustrative examples of systems that use MIR in real time for mediating multiple audio streams, such as algorithmic systems (e.g., OFFAL's *Union* and *Flock* pieces (Knotts 2016)) and live sound visualization systems (e.g., FEATUR.UX (Olowe, Barthet, Grierson, and Bryan-Kinns 2016)). The use of multiple audio input streams combined

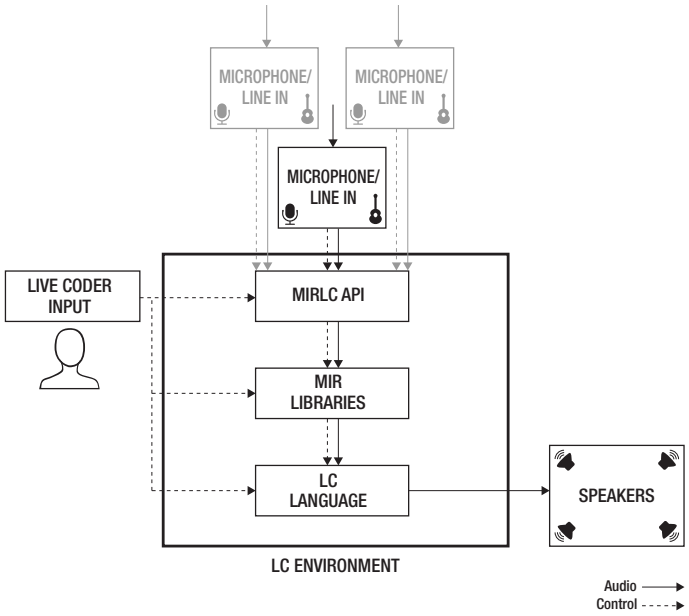



*Figure 9. Block diagram of audio remixing.*

with MIR adds complexity to the live coding environment. This approach twists the role of the live coder towards taking organizational decisions from incoming audio streams (e.g., setting volumes of the audio streams, creating mappings between audio and visual parameters), as opposed to interacting with a single audio stream and creating sounds, as in audio rewiring. The role of the live coder thus needs to be redefined, where the use of an algorithmic system to help taking live decisions in the performance space can lead to creative outcomes, as shown in the pieces *Union* and *Flock*. In the algorithmic examples, audio mixing decisions are based on features such as loudness. As shown in FEATUR.UX, there is room for more complex mappings using both low-level (e.g., RMS, spectral centroid, MFCCs) and mid-level features (e.g., peak frequency, chromagram) that are mapped to visual attributes (e.g., size, shape, color, rotation, position), which are combined with a visualization of the real-time changes.

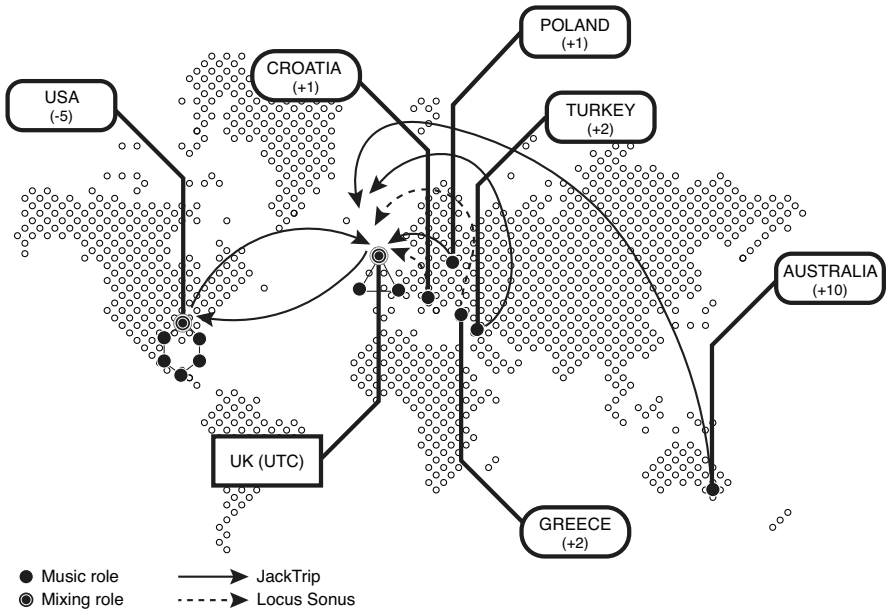



*Figure 10. Geolocation diagram of a co-joint performance between Female Laptop Orchestra (FLO) and Women in Music Tech (WiMT).*

*Prototype 3: A Case Study for Exploring Audio Remixing*

This module is designed for supporting the remix of multiple audio streams using MIR techniques in SuperCollider (online at vimeo.com/249997569). This prototype is conceptually inspired by the network music piece *Transmusicking I* (see Figure 10, online at youtu.be/AfR6UFS7Wuk), performed internationally by the Female Laptop Orchestra (FLO) and Women in Music Tech (WiMT). The lessons learned from using the web audio interface WACastMix (annaxambo.me/code/WACastMix) in this performance, as well as the experience from the audio rewiring prototype, inform the design of this prototype in terms of using feature extraction for supporting spatialization, equalization, and mixing of the incoming audio streams.

The role of LC is less obvious in this prototype, as it focuses on managing and mixing the incoming audio streams. The challenge is to use LC procedures, such as representing and manipulating the audio streams using code in real time, as well as making visible the mixing process to the audience. The use of multiple audio streams requires careful

attention from the live coder, thus visualization aids are acknowledged (similar to the graphical user interface (GUI) used in WACastMix). An interesting challenge is deciding what audio streams are performing when and how. The live coder uses functions based on feature extraction to help making egalitarian decisions (e.g., continuous vs. sporadic audio streams, vocal vs. instrumental streams), also preventing unexpected problems, such as audio streams not working momentarily. With respect to technical challenges, this approach requires a high broadband Internet connection with low latency. Also, there exists a network latency that, depending on the type of connections of the musicians who are sending audio streams, can be between 8–9 s to more than 30 s delay. This is a well-known issue in network music that affects music synchronicity (Chafe, Cáceres, and Gurevich 2010). Further explorations include combining remote and co-located audio streams, developing a GUI to support decision making, and mixing the audio streams based on creative MIR ideas, such as the multi-song mashups from AutoMashUpper (Davies, Hamel, Yoshii, and Goto 2014).

## Discussion

As shown in the literature review and illustrative examples, MIR functionality in LC environments remains quite limited compared to the state-of-the-art MIR systems outside these environments. There are several possible reasons for that. First, a surprising amount of MIR systems are not designed for real-time usage because real-time capabilities are not typical design goals of MIR researchers. Furthermore, many state-of-the-art MIR systems require calculations that take longer time than the length of the processed audio block, possibly leading to sound drop-outs and high system loads if used in a real-time context. Even if an MIR system works in real time, its latency can be too long for a given context. Second, real-time systems are often not as reliable as offline systems. Offline systems have access to more data for analysis and can search for globally optimal solutions or can use iterative approaches with undefined processing time. Third, the implementation of

MIR systems is nowadays often too complex with too many dependencies to be easily implemented as a plugin by a non-expert. Given that there is not too much overlap between researchers in MIR and LC, this impedes the integration of advanced MIR technology. Fourth, the focus of MIR researchers when designing systems is frequently on a relatively narrow group of target signals, e.g., Western pop music. This makes the systems both less usable and less appealing in the field of LC with its tendency towards experimental music. Of course, there is information that can, per definition, be not extracted in real time because systems require long-term context, for instance, for the description of music structure.

In our prototypes we have explored the extensive use of mid-level features and some workarounds to using high-level features (e.g., tag-based information). We can conclude that the usage of MIR approaches in LC is promising, yet it is still in its infancy for reasons of complexity and algorithm design, but also because of limited communication between the fields. Opening a dialog could help progressing both fields.

## Conclusion and Future Work

In this article, we have discussed MIR in live performance, focusing on the improvisational practice of live coding. In particular, we have surveyed from the literature, and explored with prototype making, three categories for using MIR techniques in live coding: (1) audio repurposing, (2) audio rewiring and (3) audio remixing. This article aims at contributing to the LC community with new approaches to using MIR in real time, as well as appealing to the MIR community for the need of more real-time systems, at the same time offering an artistic outlet and usage scenario for MIR technology. Next steps include the implementation of machine learning algorithms that can automate some tasks of the live coder and could lead to more interesting musical results that evolve over time, as well as the evaluation of these algorithms from a musical and computational perspective. The

combination of the three categories is also of future interest.

## Acknowledgments

The authors wish to thank the reviewers and editors for their insightful comments and suggestions, which have helped to improve the manuscript. We are thankful to Gerard Roma, Anna Weisling and Takahiko Tsuchiya for inspirational discussions. We appreciate the help and involvement of all the collaborators who made this research possible. We especially thank Frank Clark and Josh Smith from the School of Music at Georgia Tech for their technical support. The work presented in this article has been partly conducted while the first author was at Georgia Tech from 2015–2017 with the support of the School of Music, the Center for Music Technology and Women in Music Tech at Georgia Tech. Another part of this research has been conducted while the first author was at Queen Mary University of London from 2017–2019 with the support of the AudioCommons project, funded by the European Commission through the Horizon 2020 programme, research and innovation grant 688382.

## References

Aaron, S., and A. F. Blackwell. 2013. "From Sonic Pi to Overtone: Creative Musical Experiences with Domain-specific and Functional Languages." In *Proc. of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design*. pp. 35–46.

Bernardini, N., X. Serra, M. Leman, and G. Widmer, (editors) . 2007. *A Roadmap for Sound and Music Computing*. The S2S2 Consortium. URL https://repositori.upf.edu/handle/10230/34060.

Bogdanov, D., N. Wack, E. Gómez, S. Gulati, et al. 2013. "Essentia: An Audio Analysis Library for Music Information Retrieval." In *Proc. of the International Society for Music Information Retrieval 2013*. pp. 493–498.

Brown, A. R. 2006. "Code Jamming." *M/C Journal* 9(6). URL http://journal.media-culture.org.au/0612/03-brown.php.

Bullock, J. 2007. "Libxtract: A Lightweight Library for Audio Feature Extraction." In *Proc. of the International Computer Music Conference*, volume 2. pp. 25–28.

Chadabe, J. 1984. "Interactive Composing: An Overview." *Computer Music Journal* 8(1):22–27.

Chafe, C., J.-P. Cáceres, and M. Gurevich. 2010. "Effect of Temporal Separation on Synchronization in Rhythmic Performance." *Perception* 39(7):982–992.

Collins, N. 2002. "The BBCut Library." In *Proc. of the International Computer Music Conference*. pp. 313–316.

Collins, N. 2006. "BBCut2: Integrating Beat Tracking and On-the-Fly Event Analysis." *Journal of New Music Research* 35(1):63–70.

Collins, N. 2011. "SCMIR: A SuperCollider Music Information Retrieval Library." In *Proc. of the International Computer Music Conference*. pp. 499–502.

Collins, N. 2015. "Live Coding and Machine Listening." In *Proc. of the First International Conference on Live Coding*. pp. 4–11.

Collins, N., A. McLean, J. Rohrhuber, and A. Ward. 2007. "Live Coding Techniques for Laptop Performance." *Organised Sound* 8(3):321–330.

Davies, M. E. P., P. Hamel, K. Yoshii, and M. Goto. 2014. "AutoMashUpper: Automatic Creation of Multi-Song Music Mashups." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22(12):1726–1737.

Essl, G. 2010. "Ursound–Live Patching Of Audio And Multimedia Using A Multi-Rate Normed Single-Stream Data-Flow Engine." In *Proc. of the International Computer Music Conference*. pp. 534–537.

Fiebrink, R., G. Wang, and P. Cook. 2008. "Support for MIR Prototyping and Real-Time Applications in the ChucK Programming Language." In *Proc. of the 9th International Conference on Music Information Retrieval*. pp. 153–158.

Font, F., G. Roma, and X. Serra. 2017. "Sound Sharing and Retrieval." In T. Virtanen, M. D. Plumbley, and D. Ellis, (editors) *Computational Analysis of Sound Scenes and Events*. Cham, Switzerland: Springer International Publishing, pp. 279–301.

Freeman, J., and A. V. Troyer. 2011. "Collaborative Textual Improvisation in a Laptop Ensemble." *Computer Music Journal* 35(2):8–21.

Kirkbride, R. 2016. "FoxDot: Live Coding with Python and SuperCollider." In *Proc. of the International Conference on Live Interfaces*. pp. 193–198.

Knotts, S. 2016. "Algorithmic Interfaces for Collaborative Improvisation." In *Proc. of the International Conference on Live Interfaces*. pp. 232–237.

Latina, C. 2016. "Machine Listening Eurorack Module." Master's thesis, Georgia Institute of Technology.

Lee, S. W., J. Bang, and G. Essl. 2017. "Live Coding YouTube: Organizing Streaming Media for an Audiovisual Performance." In *Proc. of the 17th International Conference on New Interfaces for Musical Expression*. pp. 261–266.

Lee, Y. S., K. Jo, A. Weisling, A. Xambó, and L. McCarthy. 2018. "Demo Hour." *Interactions* 25(5):10–13.

Lerch, A. 2012. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. Hoboken, NJ: Wiley-IEEE Press.

Malt, M., and E. Jourdan. 2008. "Zsa. Descriptors: A Library for Real-time Descriptors Analysis." In *Proc. of the 5th Sound and Music Computing Conference*. pp. 134–137.

McCartney, J. 2002. "Rethinking the Computer Music Language: SuperCollider." *Computer Music Journal* 26(4):61–68.

McLean, A., and K. Sicchio. 2013. "Sound Choreography: Body Code." *Proc. of the Second Conference xCoAx 2014 on Computation, Communication, Aesthetics, and X (xCoAx 2013)* :355–362.

McLean, A., and G. Wiggins. 2010. "Tidal – Pattern Language for the Live Coding of Music." In *Proc. of the 7th Sound and Music Computing Conference*. URL https://zenodo.org/record/849841/files/smc_2010_067.pdf.

Olowe, I., M. Barthet, M. Grierson, and N. Bryan-Kinns. 2016. "FEATUR.UX: Exploiting Multitrack Information for Artistic Visualization." In *Proc. of the International Conference on Technologies for Music Notation and Representation*. Cambridge, UK, pp. 157–166.

Ordiales, H., and M. L. Bruno. 2017. "Sound Recycling from Public Databases: Another BigData Approach to Sound Collections." In *Proc. of the 12th International Audio Mostly Conference on Augmented and Participatory Sound and Music Experiences*. pp. 48:1–48:8.

Puckette, M. 2002. "Max at Seventeen." *Computer Music Journal* 26(4):31–43.

Puckette, M. S., T. Apel, and D. D. Zicarelli. 1998. "Real-time Audio Analysis Tools for Pd and MSP." In *Proc. of the International Computer Music Conference*. pp. 109–112.

Resnick, M., J. Maloney, A. Monroy-Hernández, N. Rusk, et al. 2009. "Scratch: Programming for All." *Communications of the ACM* 52(11):60–67.

Roberts, C., and J. Kuchera-Morin. 2012. "Gibber: Live Coding Audio in the Browser." In *Proc. of the International Computer Music Conference*. pp. 64–69.

Roberts, C., and G. Wakefield. 2017. "gibberwocky: New Live-Coding Instruments for Musical Performance." In *Proc. of the 17th International Conference on New Interfaces for Musical Expression*. pp. 121–126.

Roberts, C., M. Wright, J. Kuchera-Morin, and T. Höllerer. 2014. "Rapid Creation and Publication of Digital Musical Instruments." In *Proc. of the 14th International Conference on New Interfaces for Musical Expression*. pp. 239–242.

Rohrhuber, J., A. de Campo, R. Wieser, J.-K. van Kampen, E. Ho, and H. Hölzl. 2007. "Purloined Letters and Distributed Persons." In *Music in the Global Village Conference*. URL http://iterati.net/~rohrhuber/articles/Purloined_Letters_and_Distributed_Persons.pdf.

Roma, G., P. Herrera, and X. Serra. 2009. "Freesound Radio: Supporting Music Creation by Exploration of a Sound Database." In *Workshop on Computational Creativity Support (CHI2009)*. URL http://hdl.handle.net/10230/34579.

Roma, G., and X. Serra. 2015. "Music Performance by Discovering Community Loops." In *Proc. of the 1st Web Audio Conference*. URL http://mtg.upf.edu/node/3177.

Rowe, R. 1993. *Interactive Music Systems: Machine Listening and Composing*. Cambridge, MA: MIT Press.

Rowe, R. 2001. *Machine Musicianship*. Cambridge, MA: MIT Press.

Sanfilippo, D., and A. Valle. 2013. "Feedback Systems: An Analytical Framework." *Computer Music Journal* 37(2):12–27.

Schedl, M., E. Gómez, J. Urbano, et al. 2014. "Music Information Retrieval: Recent Developments and Applications." *Foundations and Trends® in Information Retrieval* 8(2-3):127–261.

Schnell, N., A. Röbel, D. Schwarz, G. Peeters, R. Borghesi, et al. 2009. "MuBu and Friends–Assembling Tools for Content Based Real-time Interactive Audio Processing in Max/MSP." In *Proc. of the International Computer Music Conference*. pp. 423–426.

Schwarz, D. 2007. "Corpus-Based Concatenative Synthesis." *IEEE Signal Processing Magazine* 24(2):92–104.

Serra, X., M. Magas, E. Benetos, M. Chudy, et al. 2013. *Roadmap for Music Information ReSearch*. MIRES Consortium. URL http://hdl.handle.net/10230/21766.

Sorensen, A. C. 2018. "Extempore: The Design, Implementation and Application of a Cyber-Physical Programming Language." Ph.D. thesis, The Australian National University.

Stowell, D., and M. D. Plumbley. 2010. "Delayed Decision-making in Real-time Beatbox Percussion Classification." *Journal of New Music Research* 39(3):203–213.

Tsuchiya, T., J. Freeman, and L. W. Lerner. 2016. "Data-Driven Live Coding with DataToMusic API." In *Proc. of the Web Audio Conference*. Atlanta, GA. URL https://smartech.gatech.edu/handle/1853/54590.

Wang, G. 2008. "The ChucK Audio Programming Language: A Strongly-timed and On-the-Fly Environ/mentality." Phd thesis, Princeton University.

Wang, G., R. Fiebrink, and P. R. Cook. 2007. "Combining Analysis and Synthesis in the ChucK Programming Language." In *Proc. of the International Computer Music Conference*. pp. 35–42.

Weinberg, G. 2005. "Interconnected Musical Networks: Toward a Theoretical Framework." *Computer Music Journal* 29(2):23–39.

Weisling, A. 2017. "The Distaff: A Physical Interface to Facilitate Interdisciplinary Collaborative Performance." In *Proc. of the 2017 Conference on Designing Interactive Systems*. New York, NY, USA, pp. 1365–1368.

Weisling, A., and A. Xambó. 2018. "Beacon: Exploring Physicality in Digital Performance."

In *Proc. of the 12th International Conference on Tangible, Embedded, and Embodied Interaction*. pp. 586–591.

Weisling, A., A. Xambó, I. Olowe, and M. Barthet. 2018. "Surveying the Compositional and Performance Practices of Audiovisual Practitioners." In *Proc. of the 18th International Conference on New Interfaces for Musical Expression*. pp. 344–345.

Xambó, A. 2015. "Tabletop Tangible Interfaces for Music Performance: Design and Evaluation." Ph.D. thesis, The Open University.

Xambó, A., A. Lerch, and J. Freeman. 2016. "Learning to Code through MIR." In *Extended Abstracts for the Late-Breaking Demo Session of the 17th International Society for Music Information Retrieval Conference*. URL https://s18798.pcdn.co/ismir2016/wp-content/uploads/sites/2294/2016/08/xambo-learning.pdf.

Xambó, A., G. Roma, A. Lerch, M. Barthet, and G. Fazekas. 2018. "Live Repurposing of Sounds: MIR Explorations with Personal and Crowdsourced Databases." In *Proc. of the 18th International Conference on New Interfaces for Musical Expression*. pp. 364–369.

Zils, A., and F. Pachet. 2001. "Musical Mosaicing." In *Proc. of the COST G-6 Conference on Digital Audio Effects*, volume 2. pp. 39–44.