Anna Xambó,[*] Alexander Lerch,[†] and Jason Freeman[†]
[*]Music Technology
Department of Music
Norwegian University of Science and Technology
7491 Trondheim, Norway
[†]Center for Music Technology
Georgia Institute of Technology
840 McMillan St NW
Atlanta, Georgia 30332 USA
anna.xambo@ntnu.no
{alexander.lerch,
jason.freeman}@gatech.edu

Music Information Retrieval in Live Coding: A Theoretical Framework

Abstract: Music information retrieval (MIR) has a great potential in musical live coding because it can help the musician–programmer to make musical decisions based on audio content analysis 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 live coding; when they have been used, it has been with a focus on low-level feature extraction. This article surveys and discusses the potential of MIR applied to live coding 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 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 (midlevel 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 live coding. We discuss challenges and future directions of utilizing MIR approaches in the computer music field.

Live coding in music is an improvisation practice based on generating code in real time by either writing it directly or using interactive programming (Brown 2006; Collins et al. 2007; Rohrhuber et al. 2007; Freeman and Troyer 2011). 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 process of music generation, 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 semiautomatic 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 live coding practices. Its main contributions are: a survey review of the state of the art of MIR in live coding; a categorization of approaches to live coding using MIR illustrated with examples; an implementation in SuperCollider of the three approaches that are demonstrated 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 "live coding language" and "live coding environment," which support the activity of live

Computer Music Journal, 42:4, pp. 9–25, Winter 2018
doi:10.1162/COMJ_a_00484

coding in music, will be used interchangeably in this article. Programming languages that have been used for live coding 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 and Pd (Puckette 2002), and Scratch (Resnick et al. 2009). With the advent and development of JavaScript and Web Audio, a new generation of numerous Web-based live coding programming languages has emerged, e.g., Gibber (Roberts and Kuchera-Morin 2012). Visual live coding is also an emerging practice, notably with Hydra (https://github.com/ojack/hydra) and Cyril (http://cyrilcode.com/).

Commercial Software Using MIR for Live Performance

Real-time MIR has been investigated and implemented in mainstream software ranging from: digital audio workstations (DAWs) such as Ableton Live; karaoke software such as Karaoki (http://www.pcdj.com/karaoke-software/karaoki/) and My Voice Karaoke (http://www.emediamusic.com/karaoke-software/my-voice-karaoke.html); DJ software such as Traktor (https://www.native-instruments.com/en/products/traktor/), Dex 3 (http://www.pcdj.com/dj-software/dex-3/), and Algoriddim's Djay Pro 2 (https://www.algoriddim.com/); and song retrieval or query-by-humming applications such as Midomi (https://www.midomi.com/) and SoundHound (https://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 inspire this research. There are several collaborations and conversations between industry and research looking at real-time MIR applied to composition, editing, and performance (Bernardini et al. 2007; Serra et al. 2013). To our knowledge, there has been 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 Live Coding Environments

In this section, we present a largely chronological and functional analysis of existing MIR tools for live coding 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.

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. In an earlier publication, the second author referred 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 (Lerch 2012). Serra et al. (2013) refer to tonal and temporal descriptors as *midlevel 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 et al. 2013). The distinction between low-, mid-, and high-level audio features is used here because it provides a sufficient level of granularity to be useful to the live coder.

Various audio feature extraction tools for live coding have existed for some time. In this section, we present an overview of existing tools for the three environments ChucK (Wang 2008), Max (Puckette 2002), and SuperCollider (McCartney 2002), because of their popularity, extensive functionality, and maturity. The findings and conclusions, however, are generalizable to other live coding environments. A illustrative selection of MIR tools for analysis is listed in Table 1.

Table 1. **Selected MIR Tools for Live Coding Environments**

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

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 2, the live coding environment that includes most audio features is SuperCollider, and the third-party libraries with the largest number of features are LibXtract and Freesound quark, the latter as a wrapper of the Essentia framework. Figure 3 shows the distribution of the audio features over six categories (statistical, envelope, intensity, timbre, temporal, and tonal), which are inspired by Essentia's organization (Bogdanov et al. 2013) and Lerch's (2012) feature classification.

The timbre audio features are the greatest in number (43 percent 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. The second category with stronger representation (16 percent of the total) consists of statistical audio features, with zero-crossing rate the most frequently implemented. The more commonly supported intensity audio features are loudness and root mean square (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 percent of the total), where pitch (F0) is the only feature supported extensively.

Considering these most popular features, it is noticeable that the majority are low-level (instantaneous) features (e.g., RMS or spectral centroid). These features have been widely used in audio content analysis—for example, to classify audio signals. In live coding, 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 meaningful (e.g., how to interpret continuous floating-point values of MFCCs). Midlevel features, such as onsets and tonality, can be interpreted more easily and can be used to characterize rhythmic and tonal properties, allowing 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 real-time MIR tools presented here provide the coder only with few midlevel features and show a lack of high-level features (Figure 2). As later discussed in this article, the use of high-level features (e.g., mood or genre) requires a longer time scope, and systems extracting high-level features are often prone to error (e.g., see the semantic-gap problem between low-level descriptors and semantic descriptors described by Schedl et al. [2014]). This may affect the performance negatively, compared with 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

Figure 1. Timeline of live coding programming languages and dependent MIR tools. Color version of the figure online at www.mitpressjournals.org /doi/suppl/10.1162/COMJ_a_00484.

Figure 2. Functional analysis of MIR tools for live coding: Distribution of audio features by number against low-level and midlevel audio features. Color version of the figure online at www.mitpressjournals.org /doi/suppl/10.1162 /COMJ_a_00484.

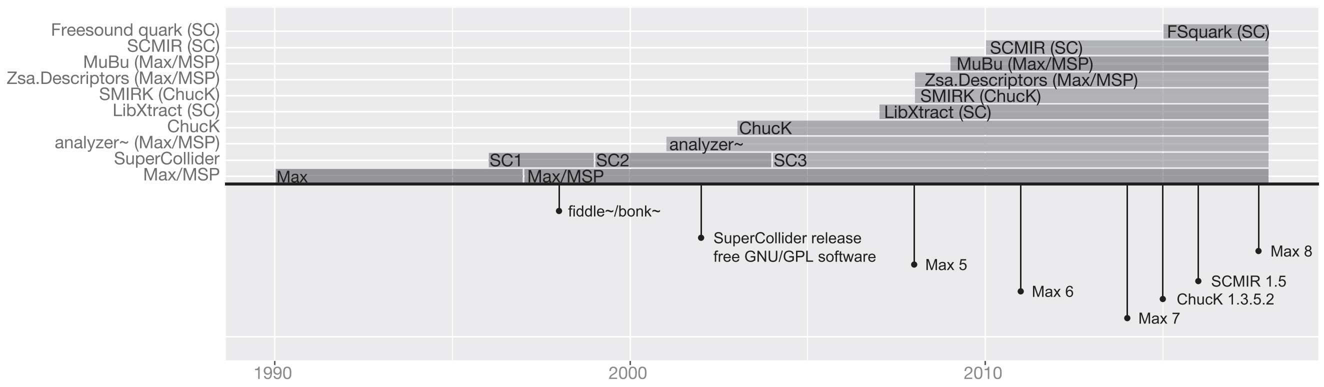


Figure 1

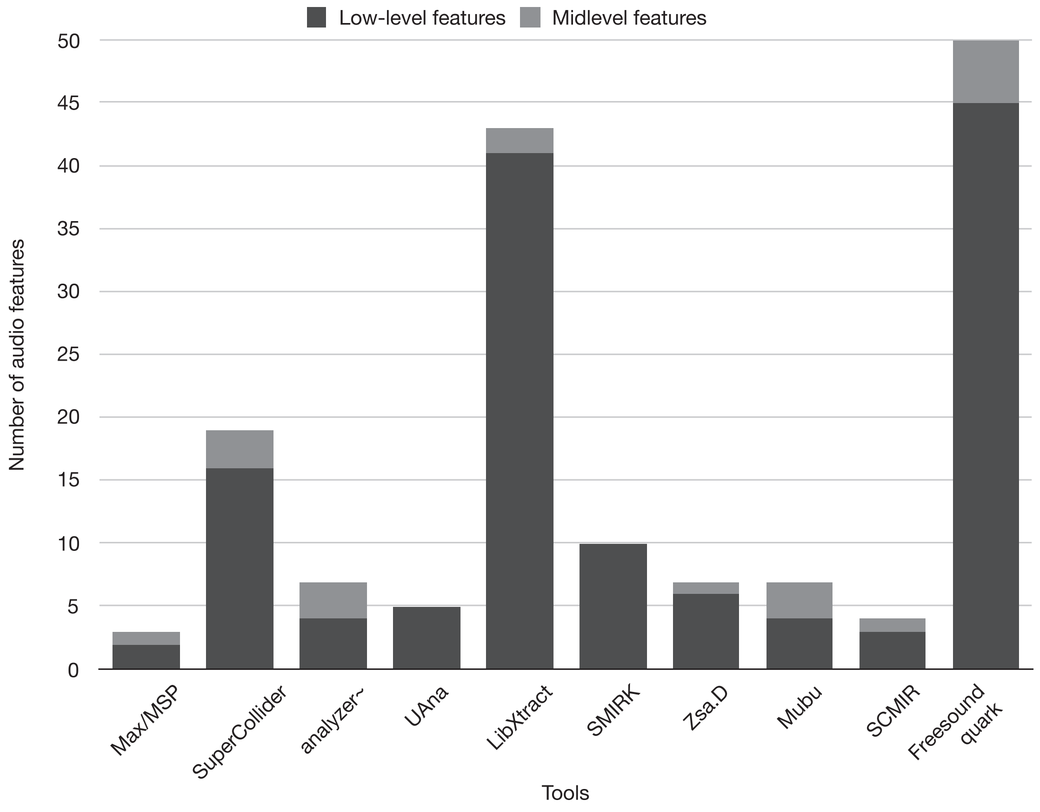
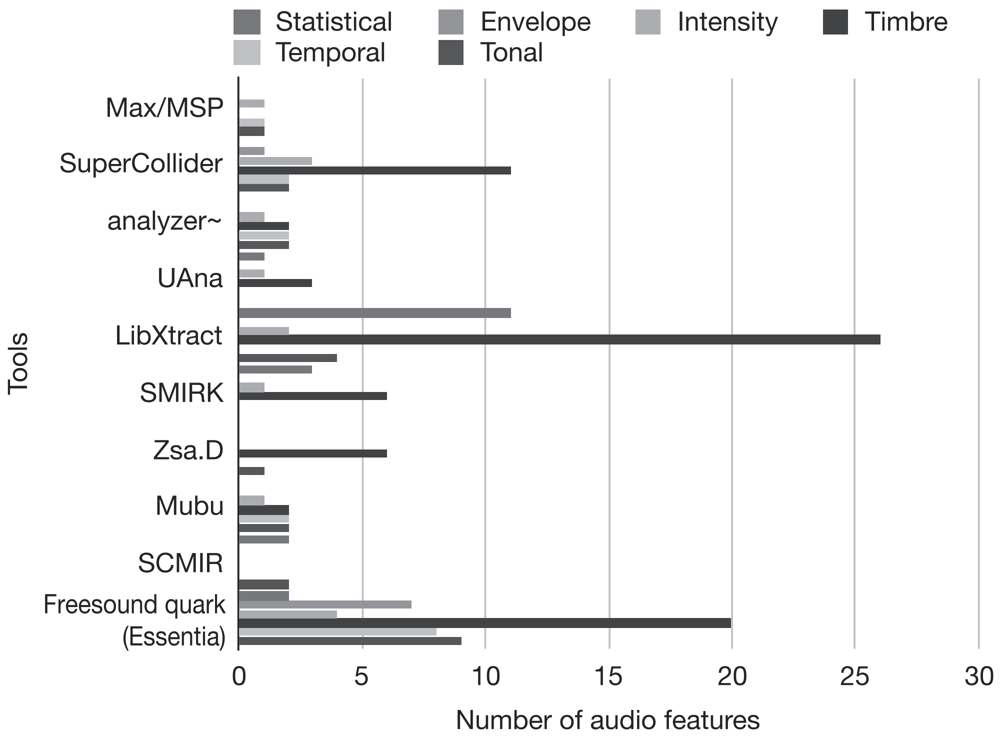


Figure 2

and musicality of the live coding session as the live coder could make decisions from a musical perspective of, for example, the overall musical form.

Figure 3. Functional analysis of MIR tools for live coding: Distribution of audio features by categories. *Color version of the figure online at www.mitpressjournals.org /doi/suppl/10.1162/COMJ_a_00484.*



Conceptual Framework on MIR in Live Coding

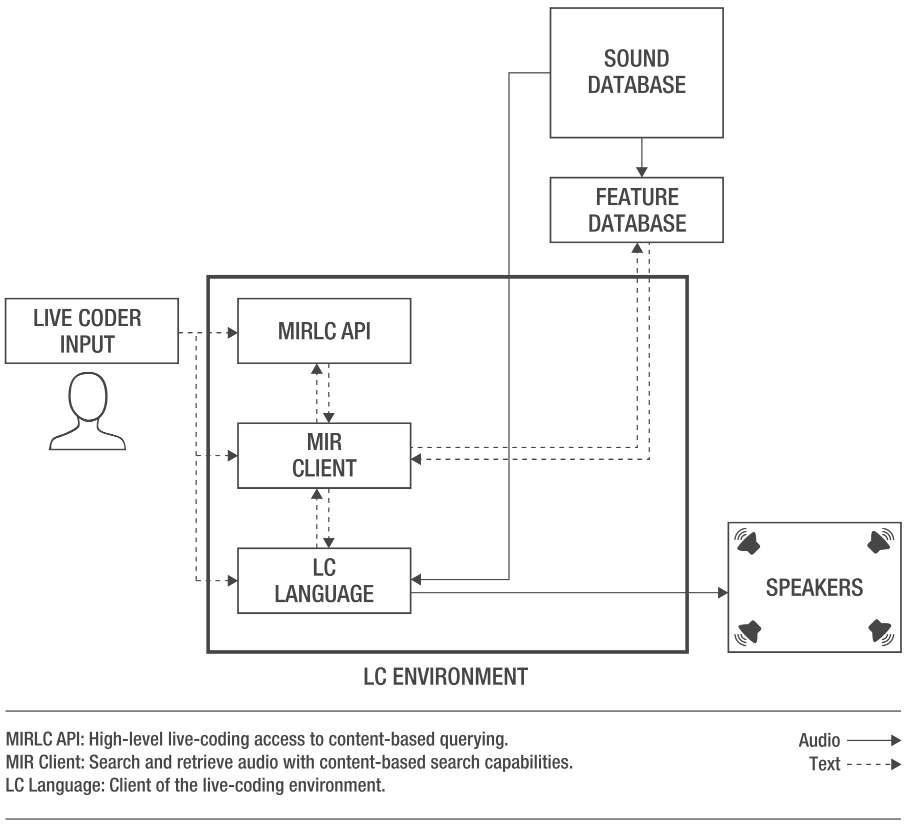
The conceptual framework presented here structures MIR approaches for live coding into three nonexclusive categories: (1) *audio repurposing*, by which we mean the analysis, retrieval, and manipulation of audio clips from a sound or music database; (2) *audio rewiring*, by which we mean the real-time analysis of an audio input signal, which can be either 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 semiautonomously.

For each of the three categories, we present a prototype. 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 constitute Music Information Retrieval for Live Coding (MIRLC, available at http://github.com/axambo/MIRLC). This is an application programming interface (API) library written in SuperCollider that is designed to provide a musical approach to using MIR techniques in live coding.

Audio Repurposing

Since the advent of the social Web, sound sharing and retrieval from online databases (e.g., ccMixter, Freesound, Looperman) have 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 identify two types of queries in audio retrieval based on features: (1) queries by range of a property, that is, retrieval of sounds that match a given interval or threshold filtered by one or multiple audio features and (2) queries by similarity, i.e.,

Figure 4. Block diagram of audio repurposing.


SOUND DATABASE
FEATURE DATABASE
LIVE CODER INPUT
MIRLC API
MIR CLIENT
LC LANGUAGE
SPEAKERS
LC ENVIRONMENT
MIRLC API: High-level live-coding access to content-based querying.
MIR Client: Search and retrieve audio with content-based search capabilities.
LC Language: Client of the live-coding environment.
Audio
Text

retrieval of similar sounds from a given example. An instance 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 mosaicking (Zils and Pachet 2001), concatenative sound synthesis (Schwarz 2007), or mashups (Serra et al. 2013) by retrieving short sounds similar to a reference. Other useful queries are by category, e.g., instrument or genre.

Figure 4 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. 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 analyze the sound database in advance and store its feature representation, thus combining the time-consuming offline analysis with ad hoc access to precomputed features and metadata from the audio clips (Serra 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 live coding 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 live coding include the retrieval of audio clips, usually by textual queries from personal or online databases, as shown in live coding sessions with

Gibber (http://youtu.be/uly7DgtfRKI?t=96) or Tidal (http://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 a preselected collection of sounds, using 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 live coding applications, include BBCut (Collins 2002), the LoopMashVST plugin (http://youtu.be/SuwVV9zBq5g), Floop (Roma and Serra 2015), EarSketch (Xambó, Lerch, and Freeman 2016), Freesound Explorer (Font, Roma, and Serra 2017), 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 midlevel 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. These systems are highly constrained to particular-use cases, however. Most of them are based on audio clips constrained to particular features, e.g., their rhythm (Floop), beat (BBCut), or timbre (LoopMashVST). During a live coding 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 or audio fragments) and long recordings (e.g., music, speech, or environmental sound scenes). Low-level features, such as MFCCs, can be used in most types of sounds (e.g., environmental sounds, speech, or music), whereas higher-level features usually make assumptions of the input (e.g., detectable onsets and pitches, identifiable tonality, or minimum length to detect rhythmic properties). As in APICultor, a live coding system should have the flexibility of filtering the results by the choice of multiple audio features. There are computational challenges related to feature aggregation that limit the length and content of this approach, however. Using an online database with preanalyzed audio features seems to be a workable approach for live coding, which can be combined with a preanalyzed local database, as discussed in previous work (Xambó et al. 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 live coding 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 (http://youtu.be/jxo4StjV0Cg). As shown in this demo, using visualization techniques to show browse-and-query processes is a compelling addition to the live coding practice, yet it broadens the definition of live coding because visual media is added to the coding environment, as explored by others (McLean and Wiggins 2010; Tsuchiya, Freeman, and Lerner 2016; Lee, Bang, and Essl 2017; Roberts and Wakefield 2017).

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 live coding using MIR, as previously discussed and assessed (Xambó et al. 2018). Figure 5 shows a test-bed performance using this module. It is designed for repurposing audio samples from Freesound using SuperCollider (online at http://vimeo.com/249968326). This module is built on top of the Freesound quark (http://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 one to browse about 400,000 sounds, either by text search, content search, or a combination of the two.

Inspired by the Web interface Freesound Radio (Roma, Herrera, and Serra 2009), this module promotes loading sounds in a musically meaningful

Figure 5. The first author live coding with MIRLC at the Noiselets microfestival, 8 January 2017, Barcelona. Color version of the figure online at www.mitpressjournals.org/doi/suppl/10.1162/COMJ_a_00484. (Photo by Helena Coll.)

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, or 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 other criteria. Each sound is played in a 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 sounds chosen at random; and an architecture to play with the groups of sounds either in sequence or in parallel. Figure 6 shows an example of the code.

In this case study, the role of live coding is to query, browse, and control the audio clips in real time. Live coding 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 of metadata with audio content analysis 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, yet unpredictable results, which demonstrates that defining musical similarity is a nontrivial task (Lerch 2012). As future work, the MIR processes

Figure 6. Example of code for audio repurposing: Code extract used for the album H2RI *by the first author (Xambó 2018).*

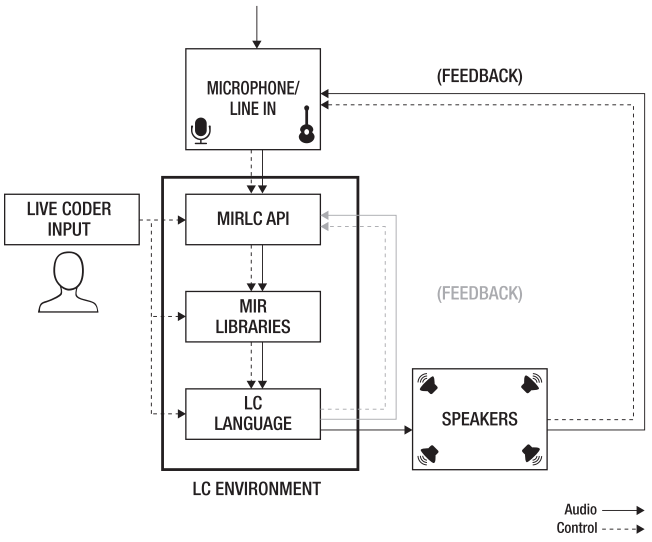
```
// Instantiation
a = MIRLCRep.new
b = MIRLCRep.new

// 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
```

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 content-based features at even higher levels (e.g., mood or genre) to be combined with text-based queries. In the current version, the use of tags has been an effective workaround (e.g., "happy," "sad," "angry," "excited," "techno," "drumbeat," or "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

Figure 7. Block diagram of audio rewiring.



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

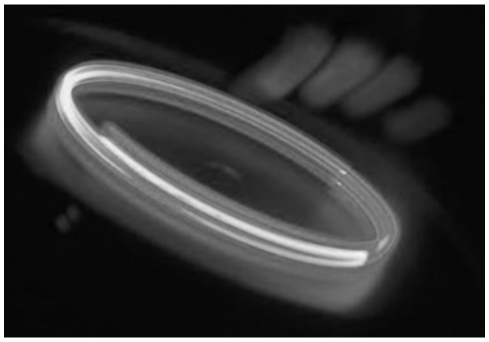
Examples of Audio Rewiring

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 Sound Choreography <> Body Code (McLean and Sicchio 2014, see also the example at http://vimeo.com/62323808). Beyond live coding, some examples combine hardware and real-time audio analysis, for instance, the Machine Listening Eurorack module (Latina 2016). Similarly, there are sound installations with feedback and real-time analysis, such as Andrea Valle's *Rumentario Autoedule* (online at vimeo.com/37148011) and Agostino Di Scipio's *Audible Ecosystems* (both installations described by Sanfilippo and Valle 2013). These examples extract numerous low-level features (e.g., spectral features) and a small subset of midlevel features (e.g., tonal and temporal characteristics). There are, however, constraints that can limit their usability. In BBCut2, for example, the audio has to be recorded before one can apply effects, which adds a delay that is hardly 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 (see, for instance, work by artists such as Sonic Arts Union, Loud Objects, and 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 or voice) in live coding using MIR. It is designed for rewiring an audio input signal as either a control signal or audio signal using MIR techniques in SuperCollider (see example online at http://vimeo.com/249997271). An early version of the prototype has been used in the piece *Beacon* (Weisling and Xambó 2018), a collaborative audiovisual work between a visual artist and an electronic musician, which has been performed internationally (Lee et al. 2018). The visual artist works with the Distaff system (see Figure 8), a customized Technics turntable (see Weisling 2017). The sound of the performer's fingers interacting with the turntable (e.g., scratching or tipping) as

Figure 8. Closeup of Anna Weisling's Distaff system, shown at the Conference on New Interfaces for Musical Expression 2017, Stengade, Copenhagen. Color version of the figure online at www.mitpressjournals.org/doi/suppl/10.1162/COMJ_a_00484. (Photo by Jimmi Brandt Fotos.)



well as the inner mechanical sounds of the device produced from these interactions, are captured with a lavalier microphone 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 the code is shown in Figure 9. The use of an audio input signal with different roles throughout the piece (e.g., audio signal or control signal) provides versatility and a wide range of variation. Live coding 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 sound source that is less percussive, for example, a voice.

The prototype presented here uncovered some conceptual drawbacks. Although a high level of synchronicity was noticed, the audience was aware of neither the role of live coding for generating sound nor the general system setup involving audio and control signals from the turntable. Furthermore, in collaborations between a live coder and a visual artist, there can be a conflict of how to make both processes visible when visuals are so prominent. Previous research has explored the extent to which both approaches (opacity and transparency) are used equally and combined in audiovisual performance (Weisling et al. 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. A future direction could be designing an environment with fewer constraints by adding more flexibility to the mappings. This would allow a greater number of spontaneous decisions to be made in real time, similar to UrSound (Essl 2010) and Gibber (Roberts et al. 2014). An interesting future challenge would be the use of higher-level audio features (e.g., mood or genre) to make decisions in real time. This would introduce some latency for accurate classification of events, so that strategies for working around processing delays would be required (for further discussison, see 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 versus decentralized or hierarchical versus egalitarian organization), roles between performers (e.g., soloist versus accompanist or sound production versus sound modification), control (e.g., shared versus individual), and types of contribution (e.g., parallel, serial, circular, or multidirectional) (Xambó 2015). This is helpful to understand the nature of network music examples and how live coding takes place using MIR on a musical network, typically applied to live remixing multiple audio streams.

Figure 10 outlines a block diagram of the audio remixing approach, where the live coder receives multiple audio streams (microphone, line in, output of the system, etc.). The real-time analysis of the audio signals is used to help the live coder taking semiautonomous decisions about the live remix.

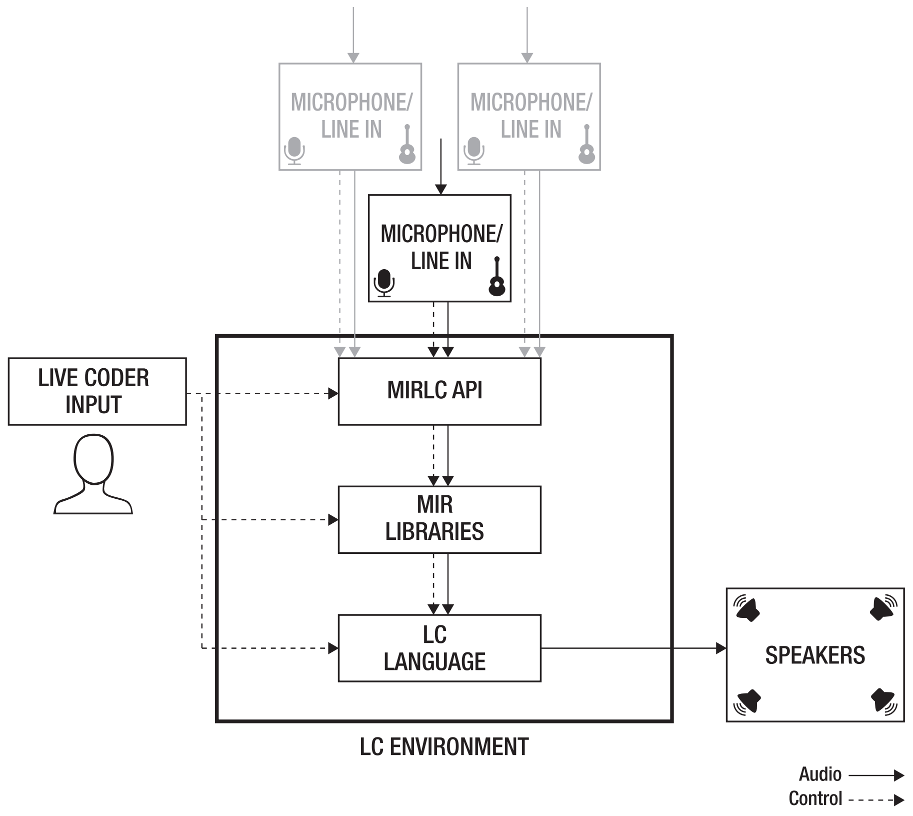
Figure 9. Example of code of the audio-rewiring prototype.

```
// 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 four 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
```

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, for instance the pieces “Union” and “Flock” by the Orchestra for Females and Laptops (OFFAL) (Knotts 2016), and live sound visualization systems, such as FEATUR.UX (Olowe et al. 2016). The use of multiple audio input streams combined 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 or 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 take live decisions in the performance space can lead to creative outcomes, as shown in the pieces “Union” and “Flock.” In the algorithmic examples, decisions about audio mixing 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, or MFCCs) and midlevel features (e.g., peak

Figure 10. Block diagram of audio remixing.



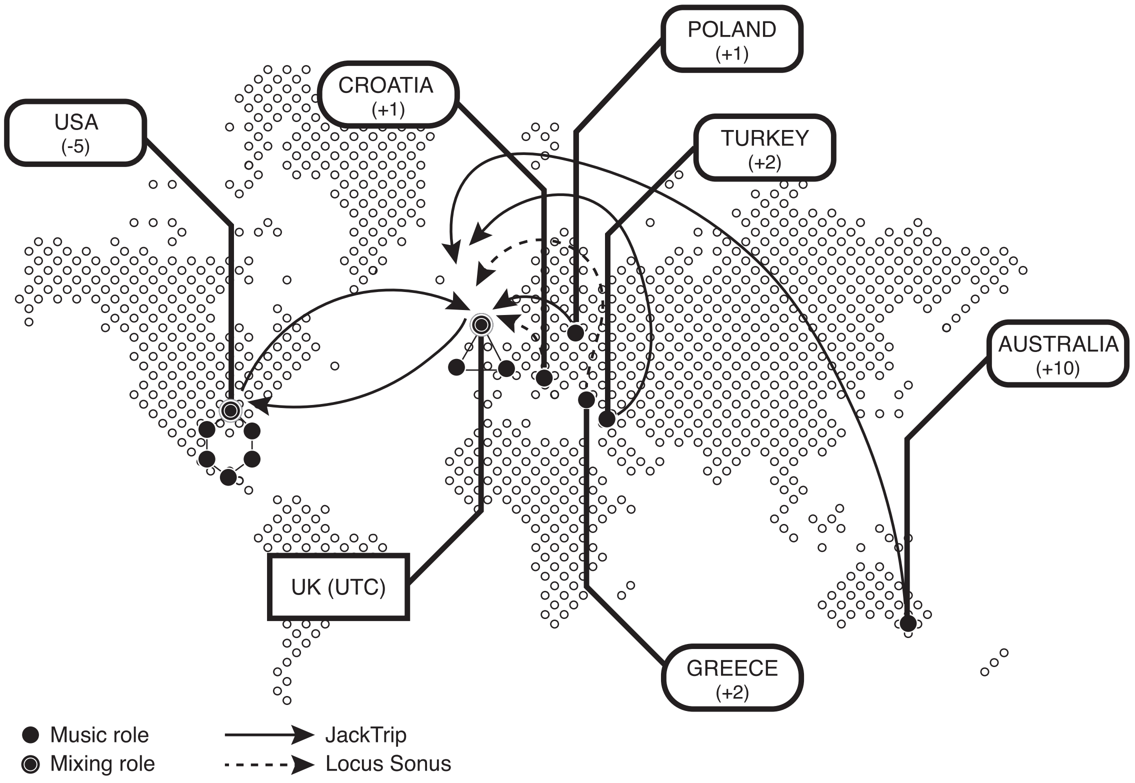
frequency or chromagram) that are mapped to visual attributes (such as size, shape, color, rotation, or position), which are combined with a visualization of the real-time changes.

Prototype 3: A Case Study for Exploring Audio Remixing

This prototype is designed for supporting the remix of multiple audio streams using MIR techniques in SuperCollider (see the example online at http://vimeo.com/249997569). This prototype is conceptually inspired by the network music piece *Transmusicking I* (see Figure 11, online at http://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 (http://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 live coding is less obvious in this prototype, as it focuses on managing and mixing the incoming audio streams. The challenge is to use live coding procedures, such as representing and manipulating the audio streams using code in real time, as well as making the mixing process visible to the audience. The use of multiple audio streams requires careful attention from the live coder, so visualization aids are acknowledged, similar to the graphical user interface (GUI) used in WACastMix. An interesting challenge is deciding which audio streams are to be performed when,

Figure 11. Geolocation diagram of a conjoint performance between Female Laptop Orchestra (FLO) and Women in Music Tech (WiMT).



and how. The live coder uses functions based on feature extraction to help make egalitarian decisions (e.g., continuous versus sporadic audio streams or vocal versus instrumental streams). This also avoids 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. There is also the problem of network latency that (depending on the type of connections of the musicians who are sending audio streams) can be between 8 and 9 sec, or even extend to more than a 30-sec 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 multisong mashups from AutoMashUpper (Davies et al. 2014).

Discussion

As shown in the literature review and illustrative examples, MIR functionality in live coding environments remains quite limited compared with the state-of-the-art MIR systems outside these environments. There are several possible reasons for that. First, a surprising number of MIR systems are not designed for real-time use for the simple reason that real-time capabilities are not typical design goals of MIR researchers. Furthermore, many state-of-the-art MIR systems require calculations that take more time than the length of the processed audio block, possibly leading to sound dropouts 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 they can use iterative approaches with undefined processing time. Third, the implementation of MIR systems is often too complex nowadays, with too many dependencies to be easily implemented as a plugin by a nonexpert. Given that there is little overlap between researchers in MIR and live coding, 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, such as Western popular musics. This makes the systems both less usable and less appealing in the field of live coding, with its tendency towards experimental music. Of course there is information that can, by definition, not be extracted in real time, because systems require long-term context, such as for the description of musical structure.

In our prototypes we have explored the extensive use of midlevel features and some workarounds to using high-level features (e.g., tag-based information). We can conclude that the use of MIR approaches in live coding is promising, yet it is still in its infancy for reasons of complexity and algorithm design, and also because of limited communication between the fields. Opening a dialog could further progress in 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: audio repurposing, audio rewiring, and audio remixing. This article aims to contribute to the live coding community with new approaches to using MIR in real time, as well as to appeal 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 article. 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 the Georgia Institute of Technology for their technical support. The work presented in this article was partly conducted while the first author was at the Georgia Institute of Technology from 2015 to 2017 with the support of the School of Music, the Center for Music Technology, and Women in Music Tech. Another part of this research was conducted while the first author was at Queen Mary University of London from 2017 to 2019 with the support of the AudioCommons project, funded by the European Commission through the Horizon 2020 program, Research and Innovation grant no. 688,382.

References

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

Bernardini, N., et al., eds. 2007. *A Roadmap for Sound and Music Computing*. S2S2 Consortium. Available online at repositori.upf.edu/handle/10230/34060. Accessed December 2018.

Bogdanov, D., et al. 2013. “Essentia: An Audio Analysis Library for Music Information Retrieval.” In *Proceedings of the International Society for Music Information Retrieval*, pp. 493–498.

Brown, A. R. 2006. “Code Jamming.” *M/C Journal* 9(6). Available online at journal.media-culture.org.au/0612/03-brown.php. Accessed December 2018.

Bullock, J. 2007. "Libxtract: A Lightweight Library for Audio Feature Extraction." In *Proceedings of the International Computer Music Conference*, vol. 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 *Proceedings 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 *Proceedings of the International Computer Music Conference*, pp. 499–502.

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

Collins, N., et al. 2007. "Live Coding Techniques for Laptop Performance." *Organised Sound* 8(3):321–330.

Davies, M. E. P., et al. 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 *Proceedings 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 *Proceedings 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, eds. *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 *Proceedings of the International Conference on Live Interfaces*, pp. 193–198.

Knotts, S. 2016. "Algorithmic Interfaces for Collaborative Improvisation." In *Proceedings 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 *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 261–266.

Lee, Y. S., et al. 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 *Proceedings 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. 2014. "Sound Choreography < > Body Code." In *Proceedings of the Conference on Computation, Communication, Aesthetics, and X*, pp. 355–362.

McLean, A., and G. Wiggins. 2010. "Tidal: Pattern Language for the Live Coding of Music." In *Proceedings of the Sound and Music Computing Conference*. Available online at zenodo.org/record/849841/files/smc_2010_067.pdf. Accessed December 2018.

Olowe, I., et al. 2016. "FEATUR.UX: Exploiting Multitrack Information for Artistic Visualization." In *Proceedings of the International Conference on Technologies for Music Notation and Representation*, pp. 157–166.

Ordiales, H., and M. L. Bruno. 2017. "Sound Recycling from Public Databases: Another BigData Approach to Sound Collections." In *Proceedings of the International Audio Mostly Conference*, paper 48.

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 *Proceedings of the International Computer Music Conference*, pp. 109–112.

Resnick, M., 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 *Proceedings of the International Computer Music Conference*, pp. 64–69.

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

Roberts, C., et al. 2014. “Rapid Creation and Publication of Digital Musical Instruments.” In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 239–242.

Rohrhuber, J., et al. 2007. “Purloined Letters and Distributed Persons.” In *Music in the Global Village Conference*. Available online at iterati.net/~rohrhuber/articles/Purloined_Letters_and_Distributed_Persons.pdf. Accessed December 2018.

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*. Available online at hdl.handle.net/10230/34579. Accessed December 2018.

Roma, G., and X. Serra. 2015. “Music Performance by Discovering Community Loops.” In *Proceedings of the 1st Web Audio Conference*. Available online at mtg.upf.edu/node/3177. Accessed December 2018.

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., et al. 2014. “Music Information Retrieval: Recent Developments and Applications.” *Foundations and Trends in Information Retrieval* 8(2–3):127–261.

Schnell, N., et al. 2009. “MuBu and Friends: Assembling Tools for Content Based Real-Time Interactive Audio Processing in Max/MSP.” In *Proceedings 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., et al. 2013. *Roadmap for Music Information Research*. MIReS Consortium. Available online at hdl.handle.net/10230/21766. Accessed December 2018.

Sorensen, A. C. 2018. “Extempore: The Design, Implementation and Application of a Cyber-Physical Programming Language.” PhD dissertation, 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 *Proceedings of the Web Audio Conference*. Available online at smartech.gatech.edu/handle/1853/54590. Accessed December 2018.

Wang, G. 2008. “The ChucK Audio Programming Language: A Strongly-Timed and On-the-Fly Environ/mentality.” PhD dissertation, Princeton University.

Wang, G., R. Fiebrink, and P. R. Cook. 2007. “Combining Analysis and Synthesis in the ChucK Programming Language.” In *Proceedings 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 *Proceedings of the 2017 Conference on Designing Interactive Systems*, pp. 1365–1368.

Weisling, A., and A. Xambó. 2018. “Beacon: Exploring Physicality in Digital Performance.” In *Proceedings of the International Conference on Tangible, Embedded, and Embodied Interaction*, pp. 586–591.

Weisling, A., et al. 2018. “Surveying the Compositional and Performance Practices of Audiovisual Practitioners.” In *Proceedings 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.” PhD dissertation, The Open University.

Xambó, A. 2018. *H2RI*. Chicago, Illinois: Pan y Rosas pyr247, compact disc.

Xambó, A., A. Lerch, and J. Freeman. 2016. “Learning to Code through MIR.” In *Extended Abstracts for the Late-Breaking Demo Session of the International Society for Music Information Retrieval Conference*. Available online at s18798.pcdn.co/ismir2016/wp-content/uploads/sites/2294/2016/08/xambo-learning.pdf. Accessed December 2018.

Xambó, A., et al. 2018. “Live Repurposing of Sounds: MIR Explorations with Personal and Crowdsourced Databases.” In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 364–369.

Zils, A., and F. Pachet. 2001. “Musical Mosaicing.” In *Proceedings of the Conference on Digital Audio Effects*, vol. 2, pp. 39–44.