Roger B. Dannenberg,[*] Nicolas E. Gold,[†] Dawen Liang,[**] and Guangyu Xia[*]
[*]School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, Pennsylvania, 15213 USA
[†]University College London
Department of Computer Science
Gower Street, London WC1E 6BT, UK
[**]Department of Electrical Engineering
Columbia University
500 W. 120th Street, Mudd 1310
New York, New York 10027, USA
rbd@cs.cmu.edu, n.gold@ucl.ac.uk,
dliang@ee.columbia.edu, gxia@cs.cmu.edu

Methods and Prospects for Human–Computer Performance of Popular Music

Abstract: Computers are often used in performance of popular music, but most often in very restricted ways, such as keyboard synthesizers where musicians are in complete control, or pre-recorded or sequenced music where musicians follow the computer's drums or click track. An interesting and yet little-explored possibility is the computer as highly autonomous performer of popular music, capable of joining a mixed ensemble of computers and humans. Considering the skills and functional requirements of musicians leads to a number of predictions about future human–computer music performance (HCMP) systems for popular music. We describe a general architecture for such systems and describe some early implementations and our experience with them.

Sound and music computing research has made a tremendous impact on music through sound synthesis, audio processing, and interactive systems. In "high art" experimental music, we have also seen important advances in interactive music performance, including computer accompaniment, improvisation, and reactive systems of all sorts. In contrast, few if any systems can claim to support "popular" music performance in genres such as rock, jazz, and folk. Here, we see digital instruments, sequencers, and audio playback, but not autonomous interactive machine performers. Although the practice of popular music may not be a very active topic for research in music technology, it is arguably the dominant form of live music. For example, in a recent weekly listing of concerts in Pittsburgh, there are 24 "classical" concerts, 1 experimental or electro-acoustic performance, and 98 listings for rock, jazz, open stage, and acoustic music.

In this article, we explore the approaches to interactive popular music performance with computers. We present a vision for such systems in the form of predictions about future performance practice. These are concretized in a reference architecture and illustrated with an example of a real system that is concerned mainly with the problem of synchronizing pre-recorded audio to live musicians.

Computer Music Journal, 38:2, pp. 36–50, Summer 2014
doi:10.1162/COMJ_a_00238

Popular Music

Categories and labels for music are risky, and the term "popular music" is particularly difficult to define precisely. Philip Tagg's tabular summary (Tagg 1982) characterizes popular music, inter alia, as produced and transmitted primarily by professionals, mass-distributed, and mainly recorded. Anahid Kassabian's (1999) discussion of popular music indicates that typically the term "popular" in this context means something opposed to an elite, but that it is not clear who or what the elite is. Derek Scott (2009) suggests that within popular music, genre is best conceived of as a category (such as blues, rock, or country), and style as a way of characterizing particular features within a genre, but acknowledges that separating these can be difficult.

Our goal is to create "virtual musicians" that can perform popular music—but what does this mean? We find many commonalities across a diverse array of music under the broad heading of popular music, including rock, jazz, folk, music theater, contemporary church, and some choral music. These commonalities dictate many aspects of our systems. In the context of this article, the term "popular music" is adopted to refer to music with these common features: organization around a steady beat and metrical structure, at least some notated parts, incorporation of improvisation, live performance, and the possibility of re-arranging sections during the performance. These features have important implications for computer music systems. There is a certain amount of circularity here: We use the expression "popular music" to determine a set of interesting system requirements, yet once we determine these, we redefine popular music to be music whose features can be addressed by our systems. Ultimately, we will achieve our goal if we can support a wide range of interactive music performance within the realm of popular music, even if that term is not well defined.

We note that our approach will not support all of what would conventionally be called popular music. For example, music with pauses or significant rubato does not fit our framework. On the other hand, our approach says nothing about harmony or tonal centers, so an atonal piece with steady tempo might be playable by our systems, even if it would not be called popular.

Turning to features and requirements of popular music performance, the main feature is a structure based on steady beats. To play popular music, an absolute requirement is accurate detection of and synchronization to beats. Because of live performance and improvisation, one cannot simply follow note sequences in a score or use triggers for synchronization. The beat is fundamental. Above the beat level, measures are important for organizational synchronization. Clapping, drumming, chord progressions, and sections are all commonly aligned on measure boundaries (or even higher levels of structure). Thus, an awareness of measures is important. Popular music also tends to be sectional, with well-defined introductions, verses, choruses, and repeats. Musicians must be aware of the structure in order to know what to play and when. This awareness comes from a combination of listening, counting measures, and visual cues. The structure may change unexpectedly in a live performance, so an important requirement is the ability to communicate improvised structural decisions during a performance.

Beyond these basic requirements lies a host of musical possibilities. We expect musicians to adjust intonation, dynamics, and style according to a variety of factors. There is a need for machine musicianship (Rowe 2001) to assist in the construction of musically interesting and stylistically appropriate performances. To cite just a few representative studies, drumming (Sioros et al. 2013), chord voicings (Hirata 1996), bass lines (Dias and Guedes 2013), and vocal technique (Nakano and Goto 2009) have all been explored and automated to some extent. Even more difficult is the problem of adjusting styles according to other musicians. For example, a piano player and rhythm guitar player should play quite differently depending on whether they play together or individually. Interactivity and responsiveness to human players is a hallmark of contemporary computer music systems, but little is known about building interactive players for popular music. For now, we will regard these possibilities as interesting for future work. Here, we focus on more basic requirements of synchronization, giving cues, and system architecture.

Computers in Popular Music Performance

Live popular music offers a wealth of opportunities for research in computing and music processing. We use the term "human–computer music performance" (HCMP) to mean the integration of computers as independent, autonomous performers into live music performance practice. In HCMP, computers become more than instruments and are seen as performers in their own right. Because HCMP is a very broad term, we add subscripts to narrow the scope; thus, HCMP_{PM} is "HCMP for popular music," and we will describe other classes of HCMP subsequently. Because it is understood that our focus here is on popular music, we will generally omit the subscript for simplicity.

HCMP will be most interesting when computers exhibit human-level musical performance, but this is such a giant advance over current capabilities and understanding that it offers little guidance for HCMP research in the short term. An alternative is to envision a future of HCMP based on realistic assumptions of machine intelligence. Thus, an important initial step in HCMP research is to imagine how HCMP systems will operate. A clear vision of HCMP will motivate further research toward this vision.

This article begins by presenting the challenges of HCMP for computer music research, posing specific problems and research directions. A reference architecture to organize the key sub-components of HCMP systems is then presented and discussed, followed by the presentation of an example HCMP-related system as a partial instance of this architecture.

A Vision for Human–Computer Music Performance

Computers have been used in music performance for many years, so before going further, we should discuss HCMP and explain how this relates to current practice in computer music (see Table 1). In general, there is great interest in autonomous performers, which have substantial potential for interesting musical applications. Popular music performance additionally requires special capabilities for synchronization to steady-beat music, so our comparison tends to emphasize synchronization characteristics. Table 1 also rates different approaches with respect to their ability to synchronize to human players, their autonomy, and their suitability to steady-beat music.

The most common use of computing in music performance is through computer instruments, typically keyboards. These, and other electronic instruments, are essentially substitutes for traditional instruments and rely upon human musicians for their control and coordination with other musicians. Because computer instruments rely on direct human control, they are not examples of HCMP. In our remaining examples, computers take on the role of performer and are therefore considered examples of HCMP.

Many composers of interactive contemporary art music use computers to generate music algorithmically in real time, often in response to live performers (Rowe 1993). These works typically take advantage of contemporary trends toward atonality and absence of a metrical pulse, which simplifies the problems of machine listening and synchronization. The problems of "playing in the right key" or "playing on the beat" are often absent. We designate this broad range of practice as HCMP_{IM} (for interactive music).

Alternatively, the practice of "computer accompaniment" (Dannenberg 1989; Raphael 2001; Cont 2008; MakeMusic 2013) offers a specific solution to the synchronization problem by assuming a predetermined score (i.e., music notation) to be played expressively by the performer while the computer follows the performer in the score and synchronizes an accompaniment. We label this work as HCMP_{SF} ("HCMP with score following").

Related work exists in the area of music-conducting systems. The work by Lee, Karrer, and Borchers (2006) is especially relevant to our work in its discussion of synchronization of beats and smooth time-map adjustment. Other work (Katayose and Okudaira 2004; Baba, Hashida, and Katayose 2010) discusses both tempo adjustment and synchronized score display using an architecture similar to some of our partial implementations. The particular problems of popular music seem to be largely ignored, however. We consider conducting-based systems to be in a separate class, HCMP_{C}.

Of course, one simple way to incorporate computers in live performance of popular music is to change the problem. The commercial sector has had a significant impact on popular music through drum machines, sequencers, and loop-based interfaces, but one can argue that popular music has adapted to new technology rather than the other way around. The precision of drum machines seems stiff, mechanical, and monotonous to many musicians, but that became the trance-like foundation of club dance music and other forms. Similarly, the inability of sequencers and other beat-based software to "listen" to human musicians has led to performances with click tracks in fixed media, or

Table 1. Interactive Music: Major Threads and Some Attributes

Class	*Description*	*Synchronization*	*Autonomy*	*Steady Beat*
Computer Instruments	Direct physical interaction with virtual instruments: digital keyboards, drums, etc.	*not applicable*	Low	*not applicable*
Interactive Contemporary Art Music (HCMP$_{IM}$)	Composed interactions; often unconstrained by traditional harmony or rhythm. Generation of algorithmic music and transformations of live performance.	Low	High	Low
Computer Accompaniment (HCMP$_{SF}$)	Assumes traditional score. Score following synchronizes computer to live performer.	High	High	Medium
Fixed Media (HCMP$_{FM}$)	Many musical styles and formats. Live performers synchronize to fixed recording.	Low	High	High
Conducting Systems(HCMP$_{C}$)	Synchronize live computer performance by tapping or gesturing beats. Best with "expressive" traditional or classical music.	Medium	Medium	Medium
HCMP for Popular Music (HCMP$_{PM}$)	Assumes mostly steady tempo and synchronization to beats, measures, and sections. Compatible with improvisation at all levels.	High	High	High

simply a fixed drum track, that live musicians must follow. We can call this practice HCMP$_{FM}$ (HCMP with fixed media). HCMP$_{FM}$ fits our definition of "independent, autonomous performer," although the level of interactivity is negligible. Ableton Live (Ableton 2009) is an example of software that uses a beat, measure, and section framework to synchronize music in live performance, but the program is not well suited to adapting to the tempo of live musicians. Robertson and Plumbley (2007, 2013) used a real-time beat tracker in conjunction with Ableton Live software to synchronize pre-recorded music to a live drummer. This extension could be considered a form of HCMP, although it does not account for the multiplicity of cue types and sectional rearrangement.

Our goal is to create an intelligent "artificial performer" that does not require a human operator sitting at a computer console, but rather uses more natural interfaces for direct control, as well as more sophisticated listening and sensing for indirect control. To develop a broader practice of HCMP, we need to imagine how humans and computers will interact, what sorts of communication will take place, and what sorts of processing and machine intelligence will be needed: a research agenda. To guide this process, we look at the practice of music performance without computers. Considering this, we construct a set of predictions that anticipate characteristics and functions of future HCMP systems (in a sense, developing requirements for such systems). These predictions will serve to guide future investigations and pose challenges for research and development. We can divide HCMP into two main activities: music preparation and music performance.

Music Preparation

"Scores" in popular music performance can range from complete and detailed common music notation (as in "classical" works) to highly abstract or incomplete descriptions such as lyrics or lists of sections. Other music representations are also common: drummers often need just the music structure (how many measures in each section) without actual instructions on what to play; keyboard, bass, and guitar players often read from "chord charts" that give chord symbols rather than specific pitches. ***Prediction 1:*** *HCMP systems will work with multiple representations of music.*

Computer-generated music in HCMP can be based on audio playback (with time stretching for synchronization), sound synthesis from MIDI event sequences, or computer composition and improvisation from specified chord progressions (for example). For many musical genres, automatic generation of parts is feasible, as illustrated by programs such as Band-in-a-Box (PG Music 2004). There are seemingly infinite varieties of styles and techniques, however, so there is plenty of room for research in this area. Many users will not have the skill, time, or inclination to play the parts themselves or to compose the parts note-by-note, so the ability to generate parts automatically is an essential feature. Users may be able to find examples of instrumental performances they like and wish to mimic, such as drum beats, bass lines, or piano accompaniments. An interesting research problem is to generate parts using musical analogies (Hofstadter 1996) to adapt examples to new harmonic or rhythmic contexts. ***Prediction 2:*** *HCMP systems will rely on stylistic generation of music according to lead sheets, in addition to pre-recorded audio and sequenced MIDI data.*

Music notation offers a direct visual and spatial reference to the otherwise ephemeral music performance. We envision capturing music notation by camera or scanner (Lobb, Bell, and Bainbridge 2005) as well as using computer-readable notation. For unstructured images, one would like to convert the notation into a machine-readable form, but like optical character recognition, optical music recognition is far from perfect, especially for handwritten (or scrawled) lead sheets. It seems essential to develop methods to annotate music images with structural information such as bar lines, repeats, and rehearsal letters (Liang, Xia, and Dannenberg 2011; Jin and Dannenberg 2013). In most cases, this annotation of music notation will be the mechanism by which the static score structure is described and communicated to the computer. ***Prediction 3:*** *HCMP systems will extend music notation to specify performance structure.*

An assumption in HCMP is that music is well structured: There are agreed-upon melodies, chord progressions, bass lines, and temporal sections such as verses, choruses, and bridges that must be communicated to all performers. If the music performance is always the same, this is trivial, but our assumption is that the structure may change, even during the performance. What happens when the vocalist decides to sing the verse again or the bandleader directs the band to skip the drum solo? Designing interfaces that are both intuitive and expressive for "programming" performances is an important problem. ***Prediction 4:*** *HCMP systems will make the relationships between scores and their performances more explicit. Terminology for specifying the location in a performance in terms of the static score will be formalized.*

One characteristic of popular music performance addressed by HCMP is the preparation of "scores" before the performance. Unlike most classical music, where the score is carefully prepared by the composer and publisher, popular music is more likely to be arranged and structured by the performing musicians. ***Prediction 5:*** *HCMP systems will provide interfaces for specifying arrangements and performance plans.*

Having discussed audio, MIDI, and various forms of music notation, it should be obvious that an important function of HCMP systems will be to provide abstractions of music structure and to allow users to integrate and coordinate multiple music media. These (and other HCMP systems) will benefit from being delivered on platforms readily available to users (Gold 2012). ***Prediction 6:*** *A primary function of HCMP systems will be to coordinate* *multiple media, both in preparation for and during live performance.*

Music Performance

HCMP must also deal with a range of issues arising from the performing ensembles themselves. The primary performance context for HCMP is a heterogeneous ensemble that, although it may be led by one member, is usually not as strongly hierarchical, for example, as in the manner of soloist and accompanist in the Western classical tradition. Individual musicians are not subservient to a leader in the generation of their own part and may at times lead themselves, leadership moving in fluid fashion among the members during a performance. Rehearsal typically establishes agreement and expectation about what is to happen during a performance with much freedom left to performance time. Rehearsal also provides opportunities to experiment with improvisations, both individually and collectively, and to select those deemed best (by an individual or the ensemble as a whole). Continued reflection on musical decisions may happen between rehearsal and performance. ***Prediction 7:*** *HCMP systems will analyze decisions made by humans in rehearsal and regenerate musical parts and strategies accordingly, following rehearsals.*

The composition and size of the ensemble may also vary between performances, and musicians may be present in performance who were not at rehearsals, causing the re-voicing or re-arrangement (in instrumental terms) of a piece prior to (or even during) performance. This may affect the content of improvisation and interaction between members of the ensemble.

Finally, the competence (in amateur ensembles especially) of individual musicians may vary widely from beginner to professional. This means that computer systems participating in a performance must be more tolerant of mistakes, planned substitutions of musical elements (e.g., different chord voicings or substitutions), and ensemble members' absence from rehearsals.

One obvious application of HCMP will be to have a computer step in to replace a missing band member. Consequently any computer "musician" taking part in such an ensemble must be capable of playing music that is appropriate (in terms of style and musical content) to the instrument for which it is stepping in, and it must do so in such a way that it blends with the ensemble. ***Prediction 8:*** *HCMP systems will need to react to the structure, style, and constitution of the ensemble in which they are performing and adapt their generative music accordingly and on the fly.*

Norms of performance practice need to be understood and respected, particularly in terms of the signals used to guide the band to different parts of the score being performed. These are often physical gestures that are either explicit (e.g., a number of fingers raised to indicate a numbered score section), or highly dependent on the local performance and temporal contexts (e.g., nodding to indicate "keep going and do that section again"). ***Prediction 9:*** *HCMP systems will be capable of responding to the physical and musical gestures used by musicians and coordinate and control their performances accordingly.*

When musicians perform together, they synchronize at several levels of a time hierarchy. At the lowest level is the beat or pulse of the music. Unfortunately, fast and accurate automatic detection of beats is not a solved problem (see Robertson and Plumbley [2013] for measurements and a discussion of the performance of some state-of-the-art live beat-tracking systems). ***Prediction 10:*** *HCMP systems will use a variety of beat-detection systems and integrate information from multiple sources in order to achieve the necessary accuracy and reliability to support computer-based "performers."*

Another level of time synchronization is the measure. Typically a group of two or four beats, measures organize music into chunks. In rock, measures are indicated by the familiar snare drum accents on beats "two" and "four" and chord changes on or slightly before beat "one." Measures are important in music synchronization because sections are aligned with respect to measures. A musician would never say "let's go to section B on the third beat of measure eight." (One might say, however, "let's go to section B and take the pickup," so we must be aware that the logical content of a measure might actually precede the notated bar line.) ***Prediction 11:*** *HCMP systems will track measure boundaries. As with beats, multiple sensors and modalities will be used to overcome this difficult problem in machine listening.*

Finally, music is organized into sections consisting of groups of measures. These sections are typical units of arrangement, such as introductions, choruses, and verses. When a performance plan is changed during the performance, it is usually accomplished by communicating, in effect, "Let's play section B now (or next)." In the case of "now," the section begins at a measure boundary. In the case of "next," the new section begins at the end of the current section. Without these higher-level temporal structures and conventions, synchronization and cues would need to be accurate to the nearest beat, perhaps just a few hundred milliseconds rather than the one to ten seconds afforded by higher-level structures. ***Prediction 12:*** *HCMP systems will be "aware" of measures and higher-level sectional boundaries in order to synchronize to human players. As with measures, multiple sensors and modalities will be used to overcome the machine-listening problem of identifying musical sections.*

In this section, we have analyzed performance practice and conventions in popular music. We have identified a set of issues and made predictions about how HCMP systems will function. We call these "predictions" rather than "requirements" because not every prediction is necessary for HCMP and realizing every prediction will require much innovation.

Reference Architecture

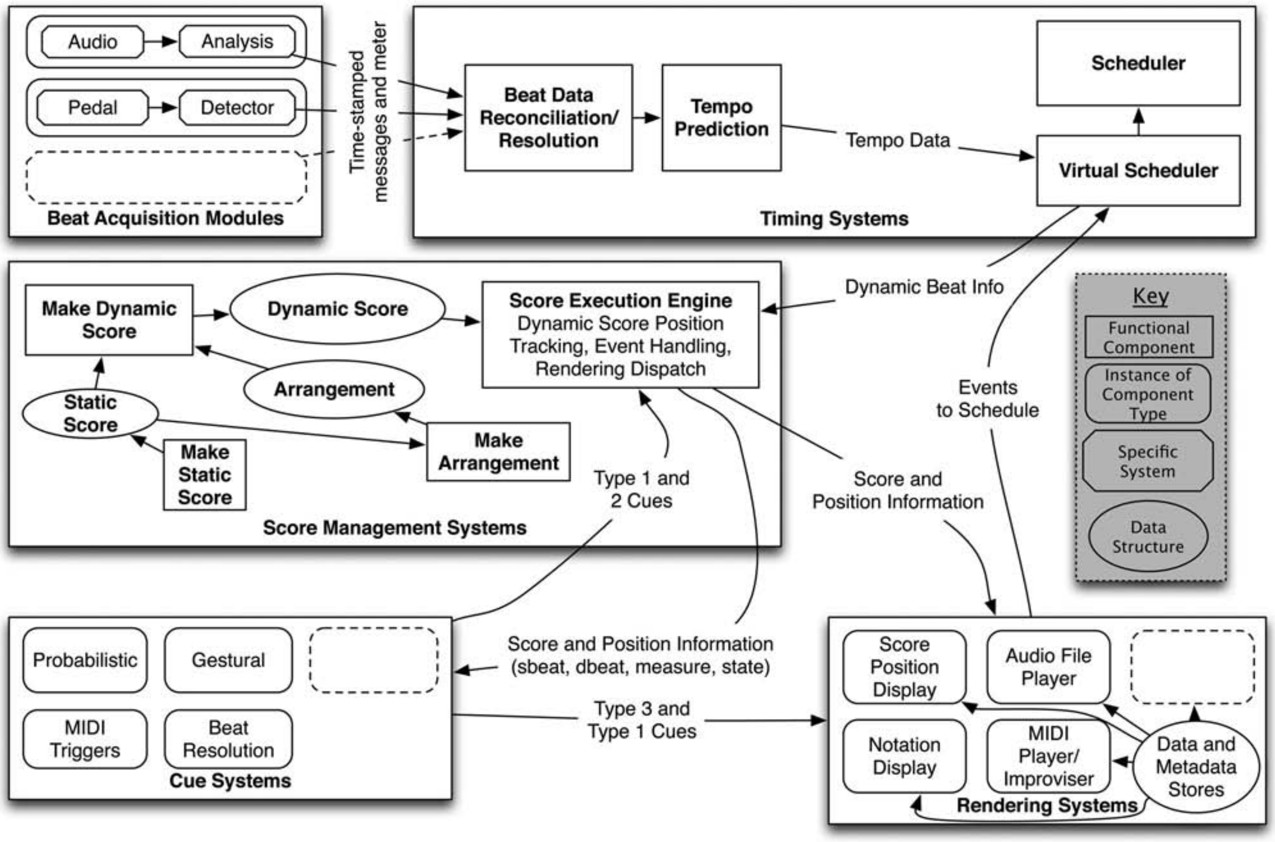
From the previous discussion, we can distill a set of general functions to be performed by HCMP systems, leading to a reference architecture. A reference architecture helps to understand and reason about components, representations, and information flows in complex systems.

Requirements

In broad terms, HCMP systems must address a range of problems including beat-tracking, tempo prediction, score-following, ensemble listening, machine musicianship, music generation and improvisation, score management, media synchronization, sound synthesis, and diffusion. Focusing on the synchronization and coordination aspects that are most characteristic of HCMP_{PM}, we require:

1. A way of representing the structure of the written score (or lead sheet or other source material) in a manner appropriate to the goal of performance (for example, elaborated measures, repeats, and other notational constructs); in other words, a static score.
2. A simple way of representing the ordering of sections of the score without needing to recreate the static score representation in full. A simple representation is required because there is typically insufficient time to fully rewrite scores in performance scenarios, the ensembles concerned may not have the expertise to rearrange music at a fine-grained level, or indeed, some of the music may exist only as memorized blocks. This is termed the *arrangement*.
3. A way in which to transform, combine, and represent the static score and arrangement together to provide look-ahead and anticipation for human and computer performers: a dynamic score. Although the internal structure of the static score sections may remain unchanged during a performance, the arrangement and, thus, the dynamic score can be rewritten to account for impromptu performance decisions (e.g., repeating the chorus an additional time). Thus, the dynamic score begins as a representation of the future unfolding of the static score and gradually becomes a history of how that score was played. The dynamic score is analogous to the execution trace of a sequential computer program.

Figure 1. A reference architecture for HCMP showing principal subsystems: Beat Acquisition derives beat times from real-time sensor data; Timing Systems estimate tempo and provide event scheduling in terms of beats; Score Management Systems combine scores, arrangements, and cues to determine what to play on each beat; Cue Systems collect map sensor data to cues; and Rendering Systems generate musical output (sounds and displays) consistent with the current beat and dynamic score. Cues convey current position (type 1), decisions about future choices (type 2), and control over voicing (type 3). See the text for further details.



4. A way in which to communicate the need for changes to the dynamic score to the performers: *cues*.
5. A way in which these representations and cues can be communicated to a range of systems involved in supporting HCMP: a *reference architecture*.

Figure 1 shows a reference architecture for HCMP systems, identifying some key subsystems required. There are several advantages to defining a baseline reference architecture. It encourages the standardization of interfaces between subsystems and components, thereby allowing many different approaches to be integrated and compared. It also promotes reasoned discussion around the appropriateness of particular components or subsystems (which may ultimately necessitate changes to the reference architecture itself). Finally, the process of defining the architecture brings to the surface issues relevant to the management of notations and representations. The architecture presented here has several components, and a companion article examines some of them in greater detail (Dannenberg et al. 2014).

Beat Acquisition and Timing Components

Real-time components are needed to keep an HCMP system coordinated with the human musicians in an ensemble. Real-time synchronization aspects are handled by components such as beat- and tempo-tracking systems (the modules for Beat Acquisition, Beat Data Reconciliation/Resolution, and Tempo Prediction in Figure 1). The Beat Acquisition modules export (at least) time-stamped messages for detected pulses and corresponding measures of confidence. Additional information, such as meter, metrical position, and tempo estimates, may also be included. Because there may be many of these systems, a reconciliation system may be needed to filter noisy beats and decide which beat-tracking source to follow on the basis of confidence and other information. This could adopt a similar approach to that outlined by Grubb and Dannenberg (1994) but accounting for the improvised nature of the music. The output of the reconciled beat data is passed to a tempo-prediction system.

Abstract-Time Components

Abstract-time components are needed to manage and schedule score events in the context of the performance. The virtual scheduler and its associated systems are concerned with aspects of abstract time in the system. The virtual scheduler should retime events that have been scheduled on a nominal time trajectory, by warping the event times according to the incoming tempo data from the tempo-prediction system. Events are then passed to an actual scheduler for real-time scheduling. This allows the unification of all media and handles the variation of latency between the various media sources in the components of the rendering system.

In Figure 1, *dbeat* (part of the Score and Position Information coming from the Score Execution Engine) denotes a monotonically increasing beat counter that serves as the "global" position within the dynamic score during a performance. The dbeat is the common, shared position used to synchronize all media. In general, the dbeat must be mapped to audio-file time, MIDI-sequence time, and other media-specific units, based on the static score, arrangement, and dynamic score.

Score Management Systems

Score management is handled by the functional components in the center box of the diagram. These systems will allow a human musician to encode, manage, and arrange scores for performance.

Cueing Systems

Cueing systems are required to allow the computer system to react to high-level structural and synchronization changes during performance (e.g., additional repetitions of a chorus). At least three types of cues are necessary.

1. *Static Score Position Cues.* These cues are necessary when synchronization with the static score is lost. Issuing a static score cue will cause the dynamic score to be recalculated accordingly.
2. *Intention Cues.* These cues are needed to inform the computer of the intended direction of the current performance (e.g., exiting a vamp section or adding an additional chorus). Issuing an intention cue (e.g., using a MIDI trigger, gesture recognition, or other method) will cause the dynamic score to be recalculated, starting at some score position as yet in the future.
3. *Voicing/Arrangement Cues.* These cues are needed to allow control over the voicing of a section. For instance, it may be desirable to prevent a particular instrumental group from playing on the first time through a repeat, but to allow them to play on the second. These cues may also control style, loudness, articulation, etc. This type of cue affects only the rendering system to which it is issued.

Rendering Systems

Rendering systems are responsible for providing multimedia output at the appropriate time. To keep the detail of the specific types of media and their output separate from the abstract architecture, each rendering system is responsible for the management of its own data (e.g., MIDI, audio, score images). Metadata is required to link these data elements to their appropriate position in the static score (and thus to the appropriate scheduling as the dynamic score is played). For example, the metadata can be a list of (*dbeat*, *position*) pairs that specify positions (sample numbers, pixels, MIDI clock number, etc.) within the data as a function of dbeats. This leaves rendering systems free to determine whether they need beat-level information or simply use the measure-level data. A score display system might map a measure to image information, or an audio rendering system might represent audio at the beat level.

Abstract beat-time information can thus be linked to real-time source material for the correct scheduling of real-time data while allowing the overall system to remain oblivious to the specific source formats being used. Rendering systems should use a callback interface whereby they schedule events with the scheduling systems. These events call the appropriate renderer at the scheduled time, causing synchronized real-time output of media in accordance with the dynamic score and beat tracking information.

Rendering systems can be as simple as a MIDI player, or an audio player with time-stretching capabilities to adjust playback tempo. Alternatively, the renderer can write or improvise parts, respond to other musicians (real or virtual), and present controls for adjusting style, timbre, and other qualities. These musically responsive renderers may require extensions to the architecture to include machine listening modules and more communication between renderers.

An HCMP Example: The Virtual Orchestra

This section describes an exemplar instance of HCMP.

Requirements

Our goal was to create a high quality, virtual string orchestra that could play along with a live jazz band in a musically convincing way. We decided to emphasize practical considerations and reliability over exotic or cutting-edge research.

One exception to this set of priorities is that we feel that HCMP must be autonomous enough to operate without a dedicated human operator. In contrast, a human could easily play string parts on a keyboard connected to a string synthesizer. This would be simple and robust, and with work might even sound good, but our objection is that it takes the entire attention of an expert musician who might otherwise play piano or guitar or some other instrument. Another problem would exist with a conducting interface (Dannenberg and Bookstein 1991; Baba, Hashida, and Katayose 2010), which would require either the addition of a conductor, or for an existing conductor to use simplified gestures to be reliably interpreted by a computer. We prefer systems that need no extra personnel to operate, yet bring new capabilities to the human ensemble.

Components

Our approach consists of several components. First, we have music representation issues: how will string parts be created, represented, and translated all the way from score to sound? Second, synchronization is critical: how will we keep the string parts synchronized to the band? Third is the quality of sound generation: how will we make convincing acoustic string sounds electronically? Finally, there is the diffusion problem: how will we organize and project string sounds into the hall?

Music Organization and Representation

The jazz standard *Alone Together*, by Arthur Schwartz, was chosen for performance, in part for its title's implicit commentary on human–computer performance. John Wilson was commissioned to arrange this piece for jazz ensemble and strings to show off the system's capabilities. The arrangement includes lush countermelodies, alternations between strings and live horns, chordal backups behind live soloists, and a pizzicato interlude with a live bass soloist.

From a computational perspective, the string parts are organized as a set of sound files. Each file has a list of time offsets corresponding to beat times, and the task of the computer system is to start playing the file at the proper time (on the proper beat), as well as to vary the playback speed so that the designated file time offsets synchronize with beats in the live performance.

We decided to implement the sound generation by recording an actual string orchestra ahead of time, to obtain a convincing sound. The sound files were recorded two or three tracks at a time in a studio, using close microphones to capture a dry sound. To create a realistic performance situation for the players, we first recorded a click track, and then recorded the actual live rhythm section (using headphones to stay with the click track). Finally, the string players played along while listening to the rhythm section over headphones. We feel that this approach gives the strings a useful rhythmic and pitch reference, avoiding any tendency to play the parts "straight" or mechanically, as might happen when playing along with a simple click track. On the other hand, the original click track reference makes it easy to identify beat times in the recordings, which is necessary for their ultimate synchronization to the live band.

The recordings were mixed to 20-track sound files with one instrument per track, comprising twelve violins, four violas, and four cellos. Each file represents a set of contiguous measures beginning at an entrance of the string ensemble and ending at a point where the entire ensemble has a rest of significant duration (at least 16 measures).

Synchronization

Synchronization requires us to begin the playback of each sound file at the proper moment and at the proper tempo, and to track the tempo and beat times of the band until the end of each file.

For simplicity, we decided to use a simple foot-tapping interface (Dannenberg 2007) to communicate beat times. Taps are in cut time (one tap every two beats) at about 85 taps per minute. We use an additional keyboard as set of triggers to cue some of the entrances.

The beat and tempo detection software interprets input according to different states. In the "initial" state, input is ignored until there are three successive taps at approximately equal time intervals. This sets an initial tempo and causes a transition to the "run" state. In the run state, the software uses linear regression over up to six previous taps to predict the next tap time. A tap that arrives within one-third of a beat period of the expected time is added to the list of beats, and a new regression is performed to update the estimated tempo and predict the next beat time. If no tap arrives during the expected time window, the system waits for a tap near the following beat. If there is no tap near this second estimated beat time, the system goes back to the initial state.

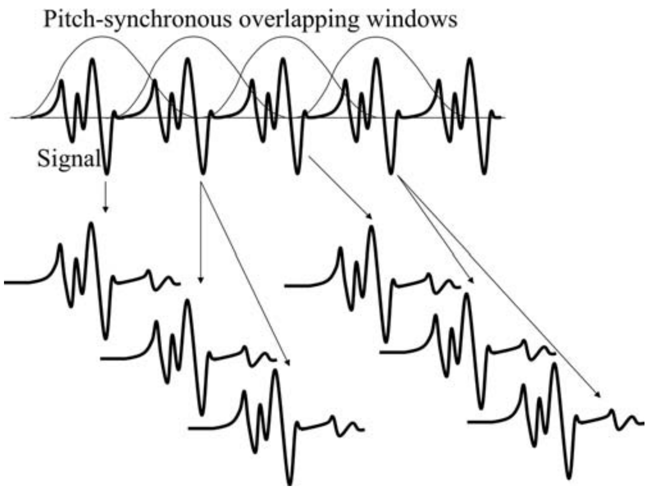
The main output of the tapping system is the linear regression of recent beats. This provides a mapping from time to beat number that can be used to schedule events in the future. As each new tap arrives, we compute a new linear regression to update the time-to-beat mapping, which is sent to a high-priority audio process. When it comes time to compute audio, the future output time of the audio (estimated by the PortAudio library) can be mapped easily to an estimated beat time and tempo as described above. This approach simplifies reasoning about timing and synchronization.

Sound Generation

Sound generation is coupled to the tapping component through the time-to-beat mapping, which is typically updated in response to each new tap. The goal of sound generation is to have the current audio output correspond to the currently estimated beat position (we treat beats as continuous, so it makes sense to say that the current beat position is 23.17, or 17 percent of the way from beat 23 to beat 24). To accomplish this, we cannot simply jump to the corresponding location in the audio file, which would create obvious and unnatural audio artifacts. Instead, we must continue a smooth playback of the audio but modulate the stretch factor to speed up or slow down in order to synchronize. We will next describe the time-stretching process and then return to the problem of synchronization.

Time stretching uses the pitch-synchronous overlap-add approach (Schnell et al. 2000). Our time stretching is mostly provided by the Elastique library from zplane.development (zplane 2007), which provides for the time stretching of a single channel of audio by a given stretch factor. The system works as follows (see Figure 2): First, audio to be processed is analyzed (offline, in our case) to detect and label pitch periods in the original audio. The labels provide not only locations and periods,

Figure 2. Pitch-synchronous overlap add (PSOLA) can be used to synchronize pre-recorded audio to live performances. Here, audio is stretched by duplicating selected pitch periods. Periods are windowed and overlapped to smooth transitions. Audio can be shortened by dropping rather than duplicating periods.



but also some spectral properties used by the stretch algorithm. At runtime, the complete analysis data are provided to the Time Stretch module, but audio is processed incrementally. To process audio, the audio stream is segmented into pitch periods, and each period is isolated through multiplication by a smoothing window, centered on the pitch period, but overlapping with adjacent periods. The windowing is organized so that if the windows are summed at their original spacing, the original waveform is recovered. To stretch the sound, windowed periods are occasionally repeated in the output (using the pitch period to determine spacing, as shown in Figure 2), thus extending the sound. To contract the sound, windowed periods are occasionally dropped. Of course, the rate of duplicating or dropping periods determines the overall stretch factor, but the algorithm has some leeway in deciding which periods to duplicate or drop. Presumably, duplicating or dropping highly periodic portions of the signal will minimize the artifacts. In practice, there are no noticeable artifacts using small stretch factors, and the most obvious giveaway is that as the stretch factor increases, vibrato begins to sound unnatural. (There is no attempt to remove and restore vibrato, but in a dense collection of 20 strings, even these artifacts will be masked.)

Our system must modulate the stretch factor continuously to track a continuously varying tempo, but tracks are stretched independently. Because stretching is not really continuous but consists of inserting and deleting whole pitch periods, we must be careful. Over time, the actual sound file position can drift away from the ideal position. A software feedback mechanism is used to measure drift and compensate through slight changes to the stretch factor (Dannenberg 2011a).

Loosely coupled to all this activity is a process that reads 20-channel sound files, de-interleaves the samples, and inserts them into first-in-first-out queues, one for each time stretcher. This allows each time stretcher to manage a slightly different stretch factor and file read position. We read data from disk in large blocks for efficiency, but use a low-priority task that can be pre-empted by a high-priority audio computation. By keeping ahead of the audio processing, we avoid blocking the audio computation to wait for a disk read to complete.

Sound Diffusion

Sound diffusion is based on multiple (eight-) speaker systems arranged across the stage. Each of the 20 input channels represents one close-miked string instrument (violin, viola, or cello). Each instrument channel is directed to only one speaker. Rather than a homogenized orchestra sound spread across many speakers, we have individual instrument sounds, each radiating from a single location and mixing in the room as with an acoustic ensemble.

Evaluation and Results

We gave one performance with our experimental system. By chance, there was an "extra" percussionist with nothing else to do in this piece, so she provided the taps. (We have used similar systems where the tapper actively performs at the same time, and a future goal is to automate the tapping.) One interesting problem occurred in rehearsal, where the tapper was naturally listening to the strings but started to tap along with them rather than the band, causing the system to drift out of synchronization. As soon as this became apparent, she began tapping with the band to correct the problem, but now the taps were falling outside of the 1/3 beat window, causing them to be ignored. This failure illustrates the subtleties of even a problem as simple as tapping beats in live performance. The public performance, however, went very well.

Musical evaluation is always difficult. Subjectively, the system maintains excellent synchronization. Laboratory simulations suggest we can predict the next beat time with an average error of less than 20 msec, although our experience tells us that average error is only part of the story and synchronization quality requires an accurate initial tempo and nearly steady tempo. Video with short descriptions can be seen online at www.youtube.com/watch?v=J_Z1GSltMPw and www.youtube.com/watch?v=R11u0S6uENA.

Although a jazz piece was performed, we did not perform it freely, and all solo sections were planned in advance. Nevertheless, the scheme for bringing in the strings on cue worked well and was demonstrated repeatedly in rehearsals. In principle, the conductor could have inserted new solos on the fly without creating synchronization problems.

Summary

None of the techniques described here (tapping, time stretching, multi-channel audio) are entirely new, but even after decades of interactive computer music it is not common to have high-quality multi-channel synchronized audio used in live performance. We are unaware of any precedent. There is even a demonstrated need for this, as seen, for example, in *Quadrophenia* performed by The Who with extensive but troublesome backing tapes in the 1970s, and the common use of click tracks on backing tapes in venues such as theme parks and cruise ships.

Conclusions

Human–computer music performance presents many opportunities for computer music research, products, and performance. We have described what we believe are the important properties of HCMP, and we have made predictions about what these systems will look like in the future. As a guide to HCMP development, we presented a reference architecture that describes HCMP systems in terms of functions, subsystems, information organization, and processing. We believe that HCMP progress is best accomplished by tackling subproblems illustrated by this architecture. Along these lines, we have begun implementing HCMP systems. In one example system, we created a live performance with cued entrances, beat-level synchronization, a high-quality virtual string orchestra (recordings of a real string orchestra, with real-time time-stretching), and multi-channel sound diffusion.

In the future, we will continue to develop and explore HCMP. We believe that there are at least three important benefits of this work. First, HCMP focuses attention on some interesting and difficult technical problems, including beat-tracking, human–computer interaction in live performance, music representation, and music generation. Second, HCMP has the potential to benefit millions of people, especially amateur musicians who might enjoy playing with virtual musicians (i.e., computer programs). Finally, HCMP capabilities offer a new creative potential to composers and performers. Even though HCMP directly addresses the needs of popular music performance, we believe that HCMP can enable creative users to develop new styles of music and performance practice that we have not yet imagined.

Acknowledgments

The support for this work by the UK Engineering and Physical Sciences Research Council (grant no. EP/F059442/2) and the National Science Foundation (grant no. 0855958) is gratefully acknowledged. Our first performance system and the music display work were supported by Microsoft Research and the Carnegie Mellon School of Music. Zplane kindly contributed their high-quality audio time-stretching library for our use. Portions of this article are based on earlier publications (Gold and Dannenberg 2011; Dannenberg 2011a, 2011b; Liang, Xia, and Dannenberg 2011).

References

Ableton. 2009. *Ableton Reference Manual: Version 8.* Berlin: Ableton. Available at downloads.ableton.com/manuals/80/ableton_live_8_manual_en.pdf. Accessed December 2013.

Baba, T., M. Hashida, and H. Katayose. 2010. "'VirtualPhilharmony': A Conducting System with Heuristics of Conducting an Orchestra." In *Proceedings of the Conference on New Interfaces for Musical Expression*, pp. 263–270.

Cont, A. 2008. "ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music." In *Proceedings of the International Computer Music Conference*, pp. 33–40.

Dannenberg, R. 1989. "Real-Time Scheduling and Computer Accompaniment." In M. V. Mathews and J. R. Pierce, eds. *Current Directions in Computer Music Research*. Cambridge, Massachussetts: MIT Press, pp. 225–261.

Dannenberg, R. 2007. "New Interfaces for Popular Music Performance." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 130–135.

Dannenberg, R. 2011a. "A Virtual Orchestra for Human-Computer Music Performance." In *Proceedings of the International Computer Music Conference*, pp. 185–188.

Dannenberg, R. 2011b. "A Vision of Creative Computation in Music Performance." In *Proceedings of the International Conference on Computational Creativity*, pp. 84–89.

Dannenberg, R., N. Gold, D. Liang, and G. Xia. 2014. "Active Scores: Representation and Synchronization in Human–Computer Performance of Popular Music." *Computer Music Journal* 38(2):51–62.

Dannenberg, R., and K. Bookstein. 1991. "Practical Aspects of a Midi Conducting Program." In *Proceedings of the International Computer Music Conference*, pp. 537–540.

Dias, R., and C. Guedes. 2013. "A Contour-Based Jazz Walking Bass Generator." In *Proceedings of the Sound and Music Computing Conference*, pp. 305–308.

Gold, N. 2012. "A Framework to Evaluate the Adoption Potential of Interactive Performance Systems for Popular Music." In *Proceedings of Sound and Music Computing Conference*, pp. 284–289.

Gold, N., and R. Dannenberg. 2011. "A Reference Architecture and Score Representation for Popular Music Human–Computer Music Performance Systems." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 36–39.

Grubb, L., and R. Dannenberg. 1994. "Automating Ensemble Performance." In *Proceedings of the International Computer Music Conference*, pp. 63–69.

Hirata, K. 1996. "Representation of Jazz Piano Knowledge Using a Deductive Object-Oriented Approach." In *Proceedings of the International Computer Music Conference*, pp. 244–247.

Hofstadter, D. 1996. *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. New York: Basic Books.

Jin, Z., and R. Dannenberg. 2013. "Formal Semantics for Music Notation Control Flow." In *Proceedings of the International Computer Music Conference*, pp. 85–92.

Kassabian, A. 1999. "Popular." In B. Horner and T. Swiss, eds. *Key Terms in Popular Music and Culture*. Hoboken, New Jersey: Wiley, pp. 113–123.

Katayose, H., and K. Okudaira. 2004. "Using an Expressive Performance Template in a Music Conducting Interface." In *Proceedings of the Conference on New Interfaces for Musical Expression*, pp. 124–129.

Lee, E., T. Karrer, and J. Borchers. 2006. "Toward a Framework for Interactive Systems to Conduct Digital Audio and Video Streams." *Computer Music Journal* 30(1):21–36.

Liang D., G. Xia, and R. Dannenberg. 2011. "A Framework for Coordination and Synchronization of Media." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 167–172.

Lobb, R., T. Bell, and D. Bainbridge. 2005. "Fast Capture of Sheet Music for an Agile Digital Music Library." In *Proceedings of the International Conference on Music Information Retrieval*, pp. 145–152.

MakeMusic, Inc. 2013. "SmartMusic: Music Education Software." Available at www.smartmusic.com. Accessed 22 October 2013.

Nakano, T., and M. Goto. 2009. "VocaListener: A Singing-to-Singing Synthesis System Based on Iterative Parameter Estimation." In *Proceedings of the Sound and Music Computing Conference*, pp. 343–348.

PG Music. 2004. "Band-in-a-Box." Victoria, British Columbia: PG Music.

Raphael, C. 2001. "Music Plus One: A System for Flexible and Expressive Musical Accompaniment." In *Proceedings of the International Computer Music Conference*, pp. 159–162.

Robertson, A., and M. Plumbley. 2007. "B-Keeper: A Beat Tracker for Live Performance." In *Proceedings of New Interfaces for Musical Expression*, pp. 234–237.

Robertson, A., and M. Plumbley. 2013. "Synchronizing Sequencing Software to a Live Drummer." *Computer Music Journal* 37(2):46–60.

Rowe, R. 1993. *Interactive Music Systems*. Cambridge, Massachusetts: MIT Press.

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

Schnell, N., et al. 2000. "Synthesizing a Choir in Real-Time Using Pitch Synchronous Overlap Add (PSOLA)." In *Proceedings of the International Computer Music Conference*, pp. 102–108.

Scott, D. B. 2009. "Introduction." In D. B. Scott, ed. *The Ashgate Research Companion to Popular Musicology*. Farnham, UK: Ashgate, pp. 1–21.

Sioros, G., et al. 2013. "Syncopalooza: Manipulating the Syncopation in Rhythmic Performances." In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*, pp. 454–469.

Tagg, P. 1982. "Analysing Popular Music: Theory, Method and Practice." *Popular Music* 2:37–65.

zplane. 2007. "Élastique Time-Stretching and Pitch-Shifting SDKs." Berlin: zplane.development. Available at www.zplane.de/index.php?page=description-elastique. Accessed December 2013.