Laptop Orchestras and Machine Learning in Real-time Music Performance

Rebecca Fiebrink[1], Perry Cook[1,2], Scott Smallwood[2], Dan Trueman[2], and Ge Wang[3]
[1]Departments of Computer Science and [2]Music, Princeton University
[3]Center for Computer Research in Music and Acoustics, Stanford University
{fiebrink, prc, skot, dtrueman}@princeton.edu, ge@ccrma.stanford.edu

INTRODUCTION

Computational support of creativity is a core concern of our daily work, as researchers and musicians working in computer music. We are enthusiastic about the prospect of attending the Computational Creativity Support workshop at CHI 2009, both to share our work on laptop orchestras and real-time machine learning in music performance, and to explore common ground with participants from creative domains, HCI, and machine learning backgrounds.

BACKGROUND OF TOPICS AND AUTHORS

PLOrk and SLOrk

All authors have been deeply involved in the Princeton Laptop Orchestra[1], or PLOrk, an ensemble of computer-based musical meta-instruments founded by Princeton professors Perry Cook and Dan Trueman in 2005. The "inherent impossibility" of pairing laptops and the traditional orchestra performance paradigm challenges us to address anew the role of technology in collaborative music-making, beyond the constraints of a standard repertoire, performance practice, or even a pre-existing understanding of how to define a laptop orchestra [9].

In PLOrk, the ensemble is comprised of undergraduate and graduate students who act as performers, researchers, composers, and software developers, exploring ways in which the computer can be integrated into conventional music-making contexts—such as chamber ensembles or jam sessions—while radically transforming those contexts. Ge Wang completed his PhD at Princeton and moved to the Stanford Center for Computer Research in Music and Acoustics, founding the Stanford Laptop Orchestra[2] (SLOrk) in the spring of 2008. We have also collaborated with colleagues around the world, who have started the Oslo Laptop Orchestra, the Boulder Laptop Orchestra (BLOrk), and others, many of them modeled after PLOrk.

The roles of the composer, conductor (which may be a human, a laptop, or neither, and is often partially or entirely algorithmically driven), performers, and software vary dramatically among our laptop orchestra pieces. For example, in *Non-specific Gamelan Taiko Fusion*[3], each laptop performer employs a graphical interface to specify the temporal positioning and instrumentation of a set of percussive sounds, a human conductor instructs the performers to choose specific instrument subsets and temporal densities, and the laptops probabilistically trigger each performer's sounds and synchronize the ensemble in time. In *Joy of Chant*[4], performers employ joysticks to continuously control parameters of a singing synthesis algorithm, and the laptops "sing" a conducted, two-part chorale, which is scored using traditional music notation. In *In/Still*[5], each performer is individually responsible for controlling sound parameters, and synthesis is also driven by sensor data collected from a human conductor/dancer in real time.

Many laptop orchestra pieces employ software written in the ChucK music programming language, designed by Ge Wang and Perry Cook [12]. The software may therefore be reprogrammed "on-the-fly" during a performance, using a practice called "live-coding" [13], which means that the algorithms and programming language themselves become real-time vehicles for musical expression and improvisation. *TBA*[6] is one piece that uses this approach.

Thus, PLOrk and SLOrk call for composers and performers to dynamically exercise algorithmic creativity at numerous interdependent levels, including the synthesis algorithms used to create sounds "from scratch," signal processing algorithms used to modify pre-recorded sounds, algorithms for creating low- and high-level musical structures, algorithms governing the relationships of laptop performers to each other (e.g., enforcing synchronization or turn-taking), and algorithms specifying the relationship of live sensor data (e.g., accelerometers held in performers' hands, or Wiimotes) to low- and high-level musical structures and synthesis. The choice of algorithms, whether they are deterministic or stochastic, the computational resources

[1] See http://plork.cs.princeton.edu/ for additional information, audio, video, and press.
[2] http://slork.stanford.edu/

[3] Video at http://plork.cs.princeton.edu/video/non-specific.mov
[4] Audio at http://plork.cs.princeton.edu/listen/winter/chant.mp3
[5] Video at http://music.princeton.edu/~jfontein/northwesternspring festival/instill
[6] Audio at http://plork.cs.princeton.edu/listen/spring/tba.mp3

they demand, the extent to which the algorithms or their parameters are controllable by human performers or a conductor in real-time, and the manner in which this control is exposed, are fundamentally both compositional and technical questions that we must address appropriately as we strive to create musical experiences that are compelling and meaningful to the performers and the audience [8]. While one avenue of work addresses these questions with regard to a general, emerging laptop orchestra performance paradigm, such work is complicated by the fact that the algorithmic choices made by composers and performers vary widely among pieces and performances, and a particular algorithmic approach may become crucial to the identity of a particular piece, performance, or software "instrument."

Another key concern in our formative and ongoing work with the laptop orchestra is the development of hardware and software interfaces that performers can easily learn to play, and that also afford musical expressivity. Cook and Trueman have been exploring musical interface design for novices and experts since many years before the formation of PLOrk [1,10].

Machine Learning for Real-Time Laptop Performance

Machine learning is a standard tool in computer-aided analysis of music. Audio analysis in particular presents a challenge without the aid of machine learning, in that there exists a "semantic gap" [7] between musically meaningful properties (e.g., notes, harmony, rhythm, instrumentation) and the low-level features that can be computed directly from audio samples (e.g., time- and spectral-domain statistics). Classification presents a useful paradigm for translating from low-level features into higher-level concepts such as instrumentation, pitch [3], or genre [11]. Classification systems may be interesting in their own right, for example to provide instrumentation or genre labels to a piece of audio, or they may be integrated into systems for music recommendation or visualization [15]. The field of music information retrieval (MIR), at the focus of the ISMIR conference, has in the past nine years come to represent the state of the art in machine learning applied to music, including audio and other types of music data.

In MIR, machine learning approaches to audio analysis are typically applied to recorded audio, and much less frequently to live audio. One barrier to the application of MIR approaches to real-time performance contexts is the paucity of general tools that support learning, audio analysis, and audio creation in real-time, within the same framework. Therefore, most systems that incorporate machine learning into real-time music contexts have been built from scratch to solve a single problem, such as real-time accompaniment of an acoustic instrument. Such systems do not provide general tools that can be modified or extended to different contexts or learning problems.

Rebecca Fiebrink and Perry Cook have a background of research in music information retrieval, particularly in the application of machine learning to audio analysis; Cook supervised some of the first work on audio genre classification [11], and Fiebrink has studied genre classification [4,2] and classification-based music recommendation and visualization systems. Fiebrink, Wang, and Cook's combined interest in music analysis and live performance has led them to recent work on creating a general platform for real-time audio analysis and applied machine learning, in the ChucK programming language [14,5,6]. Their recently released toolkit, the Small Music Information Retrieval toolKit (SMIRK)[7], provides support for real-time extraction of time- and spectral-domain features and the use of standard classifier algorithms, within ChucK. As a result, ChucK can be used to implement many standard MIR algorithms for audio classification.

Most interestingly, SMIRK allows musicians to employ classifiers in real-time—it is possible (and quite effective for many problems) to supply a classifier with labeled training examples in real-time, during a rehearsal or during a performance. For example, to create a ChucK program that produces different computer accompaniment according to whether a flute or a trumpet is playing along, a user can quickly supply training examples of both instruments upon arriving in a new performance space, so that instrument discrimination is optimized for the particular acoustics of the space. Once the classifier is trained, the user can add code to trigger the appropriate accompaniment based on classifier predictions made in real-time. The user can supply additional training examples, perhaps even during a performance, to improve classifier accuracy and compensate for unanticipated inputs, for example, a noisy audience in the background, or the use of different timbres by the instrumentalists. Furthermore, it is possible for the musician programmer to exercise control over any aspect of learning *on-the-fly*—changing classifier parameters, classifier choice, audio features, choice of classes, or even the classifier implementation itself—using ChucK's live-coding infrastructure.

Our most recent work applies on-the-fly learning to an even broader set of performance data, including performer gestures using commodity gaming controllers, custom sensors, and computer vision features extracted from webcams, as well as audio inputs. On-the-fly learning can be used to interactively build a mapping from the desired inputs to the desired synthesis parameters, allowing a composer or performer to construct a new "instrument" to play computer music. During the performance of the composition *nets 0*[8], performers interactively train neural networks to build increasingly complex and personalized

[7] Code and examples at http://smirk.cs.princeton.edu/

[8] Video at http://www.cs.princeton.edu/~fiebrink/nets0/

instruments using the controller of their choice (including joysticks, webcams, and custom sensors).

In summary, we believe that "on-the-fly learning" is a compelling new way to build interactions between computers and human performers, allowing a computer to base its actions on whatever high-level, musically meaningful concepts that human musicians decide are relevant at any given moment.

TOPICS FOR EXPLORATION

In this section, we enumerate several topics that arise at the heart of our work with laptop orchestras and real-time machine learning. Our current technical and creative work explores some of these issues, and we are very interested in how other workshop attendees confront similar issues, and in what insights others may offer from their expertise in more traditional machine learning and interface design domains.

Real-time Interaction with Algorithms

We are not interested in "algorithmic composition" in the traditional sense, which leaves little creative agency to the human after a generative algorithm has been constructed, but rather in real-time performance wherein humans apply their expertise and creativity through their interactions with software. In our music, the composer might specify the framework within which humans exercise control, including the algorithms used at different levels of music creation, the parameters that are exposed to users, and the interfaces through which users can control these parameters. The conductor and performers might then create music by expressively acting within this framework over the performance of the piece. "Creators" and "users" may have analogous roles with respect to algorithms in other domains. Therefore, a set of general questions regarding computational creativity in real-time domains includes:

- What commonalities exist in issues of interface design for real-time creative systems? We define "interfaces" broadly, including software, hardware, and programming languages. Issues of interest include the design of methodologies for characterizing and evaluating interfaces, designing interfaces for varying degrees of task specificity (e.g., designing an "instrument" that can be played in different styles and contexts, versus a "piece" that is played just one way), and designing interfaces for users with varying expertise. The International Conference on New Interfaces for Musical Expression, or NIME, treats this topic in the domain of music, but we know of no venue to explore this across disciplines.
- What commonalities exist in the process of *creating* interactive, real-time experiences (analogous to the musical composition and software design processes for musical performance)? What tools and techniques do others use in their respective domains?
- What fundamental HCI techniques may be of use in characterizing and designing algorithms for real-time interactive contexts? For example, to what extent might we talk about the *affordances* of algorithms in such contexts? Different algorithms may accomplish similar computational goals (e.g., solving an equation, producing a trained classifier) but afford different control parameters and interaction styles to a user, particularly in a creative, real-time context. Algorithmic distinctions may significantly impact the user experience regardless of any overlying user interface (whether that interface is a GUI, live-coding programming language, or other). For example, classifiers such as AdaBoost or support vector machines require a distinct training phase to produce a prediction rule, whereas "lazy" learners such as k-nearest-neighbor do not. In our experience, when employing a classifier to learn a concept "on-the-fly," this distinction can become important; it may be that a user prefers to use a lazy learner even at the expense of lower accuracy and slower predictions, in order to forego the wait for classifier training each time he or she adds more examples and observes classifier performance. That is, the user experience is impacted by aspects of algorithms that are not of concern to standard analysis (e.g., time complexity or accuracy bounds); this suggests that we might draw on HCI principles to analyze and augment standard algorithms, and even design new algorithms, for real-time interactive contexts.

Real-time and On-the-Fly Learning

On-the-fly, real-time learning is of particular interest to us for several reasons. As discussed in the previous section, it gives us a new perspective on standard algorithms, applied in a new context with constraints and goals very different from those the algorithm designers may have had in mind. An ideal on-the-fly, real-time learning interface might have the following properties:

- The user has access to timely and meaningful feedback about a classifier's performance (e.g., how long a training round might take, what sort of errors the classifier is likely to make), so that he/she may employ this information to make rational choices regarding the classifier and parameters.
- The user has the ability to dynamically exercise meaningful control over the classification process, for example by specifying limits on training time in exchange for a tradeoff in accuracy, providing additional training data and re-weighting training examples to fine-tune classifier performance, and providing on-line feedback on classifier performance, all using an interface that is appropriate for real-time use.
- Both expert and novice users are able to interact effectively with the system, regardless of the extent of their machine learning knowledge.

Some of the above properties could be supplied by novel user interfaces for visualization and control, some suggest enhancements to existing algorithms such as the addition of new parameters, and some may suggest new algorithms. Certain ideas are reflected in current theoretical work in machine learning (e.g., work on on-line and adaptive learning, or learning with concept drift), and we'd be excited to talk with participants with more traditional machine learning backgrounds who might see connections with current research. We would also like to engage with other machine learning practitioners working in creative domains who are confronting similar ideas, or who are interested in exploring them more deeply.

Interfaces and Toolkits

Finally, we have a general interest in interacting with others who develop software and hardware interfaces for real-time, creative human-computer interactions. We will be happy to discuss and/or demonstrate hardware and software interfaces and toolkits that we have made for music performance. Furthermore, several questions may be very interesting to address in a cross-disciplinary manner, including: How are interfaces for real-time, creative tasks unique from non-real-time and non-creative tasks? To what extent can we use standard toolkits and methodologies, and to what extent must we create our own? How can interfaces best support expressivity? improvisation? How can interfaces support real-time creative collaboration among humans, in both performative contexts (such as the laptop orchestra) and offline contexts (such as collaborating to compose new pieces)? What role should learning and adaptation play, if any, in creating interfaces that are customized to individual users possessing different capabilities and levels of expertise?

COLLABORATIONS AND INSPIRATIONS

In summary, we welcome the opportunity to collaborate and brainstorm with people experimenting with real-time learning and/or live coding in other domains, people who love designing new learning algorithms for unique domains and seeing them applied, people who are interested in design aspects around live performance hardware and software, and people who like thinking deeply about HCI aspects of algorithms and machine learning in real-time, creative contexts. In turn, we offer ourselves as a group of people with extensive experience in computer music performance and composition, music interface design, and applied machine learning, and who have strong ties to the music-specific research communities in these areas.

REFERENCES

1. Cook, P., "Principles for designing computer music controllers," *ACM CHI Workshop in New Interfaces for Musical Expression (NIME)*, 2001.
2. DeCoro, C., Z. Barutcuoglu, and R. Fiebrink, "Bayesian aggregation for hierarchical genre classification," *Proc. International Conference on Music Information Retrieval (ISMIR)*, 2007.
3. Ellis, D. P. W., and G. E. Poliner, "Classification-based melody transcription," *Machine Learning* 65:439–56, 2006.
4. Fiebrink, R. "An exploration of feature selection as a tool for optimizing musical genre classification," MA Thesis, McGill University, 2006.
5. Fiebrink, R., G. Wang, and P. R. Cook, "Foundations for on-the-fly learning in the ChucK programming language," *Proc. International Computer Music Conference (ICMC)*, 2008.
6. Fiebrink, R., G. Wang, and P. R. Cook, "Support for MIR prototyping and real-time applications in the ChucK programming language," *Proc. International Conference on Music Information Retrieval (ISMIR)*, 2008.
7. Lew, M. S., N. Sebe, C. Djeraba, and R. Jain, "Content-based multimedia information retrieval: State of the art and challenges," *ACM Trans. Multimedia Computing, Communications, and Applications*, 2(1): 1–19, 2006.
8. Smallwood, S., D. Trueman, G. Wang, and P. Cook, "Composing for laptop orchestra," *Computer Music Journal,* Spring 2008, 32(1): 9–25.
9. Trueman, D., "Why a laptop orchestra?" *Organised Sound*, 2007, 12(2): 171–9.
10. Trueman, D., and P. Cook, "BoSSA: The deconstructed violin reconstructed," *Journal of New Music Research* 29(2), 2000.
11. Tzanetakis, G., and P. R. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech and Audio*, July 2002.
12. Wang, G., and P. R. Cook, "ChucK: A concurrent, on-the-fly audio programming language," *Proc. International Computer Music Conference (ICMC)*, 2003.
13. Wang, G., and P. R. Cook, "On-the-fly programming: Using code as an expressive musical instrument," *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, 2004.
14. Wang, G., R. Fiebrink, and P. R. Cook, "Combining analysis and synthesis in the ChucK programming language," *Proc. International Computer Music Conference (ICMC),* 2007.
15. West, K., S. Cox, and P. Lamere, "Incorporating machine-learning into music similarity estimation," *Proc. ACM Multimedia,* 2006.