2012

Cross-talk: A Shared Parameter Space for Gesturally Extended Human/Machine Improvisation

William Brent
American University

Adam James Wilson
CUNY New York City College of Technology

CROSS-TALK: A SHARED PARAMETER SPACE FOR GESTURALLY EXTENDED HUMAN/MACHINE IMPROVISATION

William Brent

Assistant Professor, American University
Washington, D.C.
w@williambrent.com
http://www.williambrent.com

Adam Wilson

Composer & Software Engineer
New York, NY
ajw@adamjameswilson.info
http://www.adamjameswilson.info

ABSTRACT

This paper describes *Cross-talk*, a piece of music and performance system for two instruments augmented with infrared motion-tracking capability, and an artificial software improviser. *Cross-talk* was commissioned by the Ammerman Center for Arts and Technology at Connecticut College, for the 13th Biennial Symposium on Arts and Technology. The work is part of an ongoing collaboration focused on developing integrated hardware and software performance systems to extend the timbral and expressive capabilities of traditional musical instruments and to generate musical structure in response to information retrieved from human performers in real-time. Artistic motivations and prior related work are presented here, along with a summary of the programmatic narrative behind *Cross-talk* and an accompanying qualitative description of the piece. Technical details are provided for important components of the work, including the Gesturally Extended Piano and the "factorOracle" software module, which is used to facilitate the system's machine improvisation capability.

1. MOTIVATIONS

Cross-talk is the most recent manifestation of a collaboration that developed over the course of our work at the Center for Research in Computing and the Arts, part of the California Institute for Telecommunications Technology at the University of California, San Diego. It began with a shared interest in technology-based methods for extending the capabilities of traditional musical instruments and performance practices. At the time, William was building MIDI-controlled percussion robots and writing real-time audio analysis software, and Adam was developing software for automatic composition. Both of us were involved in improvised music, but still very much connected to the tradition and technique of processing audio and music information from performances of composed music. We were drawn to the challenges and opportunities associated with applying these same real-time techniques to improvised performances of unconventional instruments—a context in which fewer constraints exist when mapping control information to sound manipulation processes.

In 2009, we undertook a project to explore these ideas, constructing a guitar-driven "hyperinstrument" that offered new timbral and musico-structural possibilities to the improvising guitarist. Specifically, it allowed the performer to launch and modify real-time compositional processes realized by a collection of synthesizers and thirty-two percussion playing robots called "ludbots"[1]. The hyperinstrument was given the title *Duoquadragintapus*, after the sum of digits made available to the ordinarily ten-fingered guitarist.

2. PRIOR RELATED WORK

2.1. The Duoquadragintapus

The *Duoquadragintapus* was performed at the California Institute for Telecommunications Technology at the University of California, San Diego, and is documented in *Automatic Improvisation: A Study in Human/Machine Collaboration* [8]. The most important aspect of the software component of the *Duoquadragintapus* is its ability to improvise streams of pitches or rhythms that relate to what has been played by the human performer. This capability was realized by implementing a Max[2] external for real-time construction and traversal of a factor oracle automaton. The external, called simply "factorOracle," also plays a major role in *Cross-talk*, and is described in detail in section 4.2.

In general terms, the factor oracle is a type of finite state automaton, or an abstract model of a machine with a quantifiable number of states and transitions between states. In the *Duoquadragintapus*, states are represented by points between sequences of notes played by the guitarist, and transitions are represented by the notes themselves. The factor oracle allows us to traverse the sequence of notes originally

[1] http://williambrent.conflations.com/pages/projects.html#ludbots
[2] http://cycling74.com

played by the guitarist and "jump" from one subsequence of notes to another, using common subsequence endings to guide the transitions. Depending on how we navigate the automaton, we can produce note sequences that are very similar to or very different from the original string of notes. In either case, the output of the oracle can be viewed as an improvisation produced by "listening" to the performer.

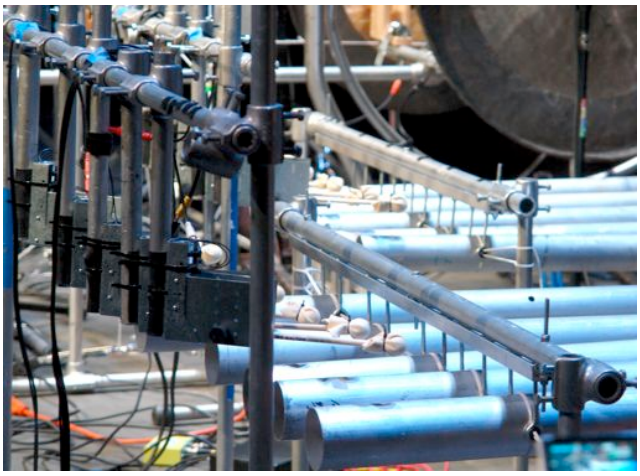

Figure 1. Percussion-playing "ludbots" onstage in the *Duo-quadragintapus*.

2.2. DILib and Motion Capture

The infrared (IR) motion capture system used in *Cross-talk* is an important performative element. Its associated hardware and software enables the Gesturally Extended Piano described in section 4.1, as well as gestural extensions to the guitar. The tracking system follows the movements of reflective markers attached to the instrumentalists, and outputs normalized three-dimensional position data. Its software component was built using the IR blob tracking module from DILib [4]—a set of abstractions and externals for the Pure Data[3] programming environment.

Initially developed as a teaching tool for a course on digital musical instrument design, DILib (the Digital Instrument Library) is intended to streamline the process of realizing musical instruments that make use of built-in laptop hardware, accelerometers, infrared fingertip tracking, full body tracking, multitouch surfaces, and other control data streams. In addition to providing convenience, the library's components are designed to establish a level of standardization with respect to the varied methods for obtaining sensor data from widely available hardware.

IR blob tracking is used to achieve different ends for each of the hyperinstruments in *Cross-talk*, but in both cases, higher level information beyond the raw coordinates of tracked points is extracted. For example, in addition to the position of the pianist's right hand, the software reports the distance between the hands, the velocity of the hands, the angles formed between tracked points, etc. Once this interdependent network of information is mapped to synthesis parameters, tendencies and idiomatic characteristics comparable to those associated with acoustic instrument parameters can emerge.

3. CROSS-TALK

Cross-talk is inspired by one of the first electronic instruments, the Telharmonium, invented by Thaddeus Cahill in 1897. The instrument was essentially a collection of electric motors used to generate alternating currents in the audible frequency range. The various motors, or "dynamos," could be combined to produce more complex timbres, arguably making the Telharmonium the first electronic additive synthesizer. The 200-ton machine lived most of its life, in various manifestations and locales, in New York. The final version was built in 1911, at which time a plan was formulated to transmit music from the Telharmonium through the telephone system to homes and businesses throughout the city. The plan failed, largely because signals from the Telharmonium produced unwanted cross-talk in the telephone system, and conversations had the potential to be interrupted by sometimes bizarre-sounding music (Cahill had employed a 36-tone-per-octave system in an attempt to approximate just intonation) [5], [6, p.108], [7].

Cross-talk presents an analogy to the disillusion of the Telharmonium, and reflects on the ever-evolving nature of technology as well as the decline and resurgence of aesthetic ideas expressed through technological media over time. The work defines a conversation between improvisors, taking place through a complex network of technological media—a conversation often interrupted or transformed in unpredictable ways consequent to the design of the network itself.

Each improvisor in *Cross-talk* plays a hyperinstrument, formally defined as "a musical instrument designed or adapted to be used with electronic sensors whose output controls the computerized generation or transformation of the sound"[4]. Hyperinstruments can be made to augment the timbral space of traditional instruments, as well as provide interfaces to complex generative musical systems. Among other things, such interfaces allow musicians to call preprogrammed compositional algorithms with arguments supplied by musical data generated in live performance.

The piano and the fretless electric guitar provide the traditional bases for the hyperinstruments developed in this work. The piano system, called the Gesturally Extended Piano (GEP), is a hyperinstrument that tracks the pianist's arm motions in order to control real-time synthesis and processing of the piano's acoustic sound. The guitar system, called,

[3] http://puredata.info

[4] Oxford English Dictionary

analogously, the Gesturally Extended Guitar, tracks the instrument's headstock in three-dimensional space. Position data is used to affect changes in the guitar's output signal and to provide input to a virtual improvisation partner. The virtual improviser—implemented in Max, and built around the factorOracle external—also collects and analyzes other features of the performance, including pitch and note-onset information, and repurposes them as inputs to a collection of generative music algorithms.

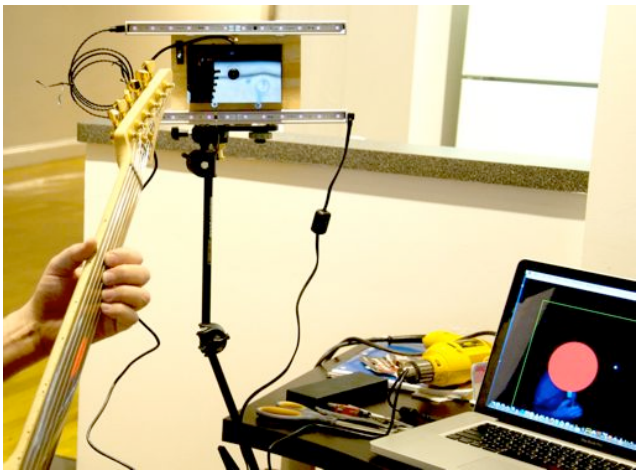

Figure 2. Infrared motion-tracking hardware for the guitar.

As each player attempts to control his own hyperinstrument, a potential side effect may occur: every predictable change one improvisor induces in his own signal—through movement—can be made to produce a parallel unpredictable effect on the signal generated by the other improvisor. This results in a very real "cross-talk" in the musical system, which the performers must address recursively.

A further analogy to the telephonic interruptions of the Telharmonium is found in the occasional and random decoupling of the performers from the generative and transformative musical processes that make up their hyperinstruments. In these moments, the software improvisation component—ordinarily working off of musical information from the guitar, and potentially modified by control streams generated by the GEP—begins to produce sounds completely unrelated to the musical conversation at hand. Again, the human performers must struggle to make sense of these interludes, succeeding or failing to smoothly incorporate them into the dialogue.

4. IMPLEMENTATION DETAILS

4.1. Gesturally Extended Piano

The GEP is an augmented instrument controller that tracks performer movements in order to steer real-time audiovisual processing and synthesis. All required hardware—including a high frame-rate USB camera fitted with a band pass filter, an IR light array, a camera mounting arm, and spherical reflective markers—has been chosen with an emphasis on widespread availability and relatively low cost.

With certain limitations, the use of IR light drastically simplifies the problem of following specific objects within a complex scene. IR blob tracking has been used as a reliable means of capturing motion information in a variety of contexts. The basic method is to shine a particular wavelength of IR light on a scene, and place highly reflective markers on key points of a moving body. Near the light source, a camera fitted with a bandpass filter tuned to the same IR wavelength observes the scene. Frames in the digital video stream are then subjected to some basic preprocessing before being fed to a blob tracking algorithm.

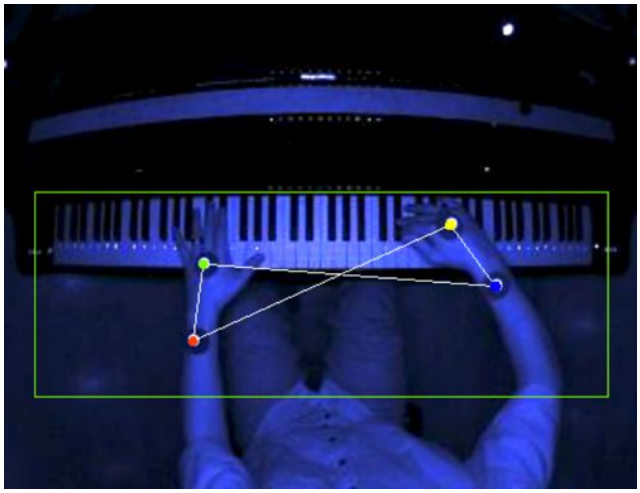

Figure 3. Overhead view from the GEP tracking system.

In the case of a pianist, the most elementary pieces of movement information are the positions and angles of the forearms in relation to the keyboard. This information can be obtained by following two key points on each arm, allowing motions that extend relatively naturally from standard piano technique—such as flexing of the wrists and rotation of the forearms—to be used for modulating given synthesis parameters. Among other possibilities, augmenting the piano via motion tracking allows for intuitive control over sound characteristics that are usually inaccessible when playing the piano, such as continuous changes in pitch and volume.

Though none of the control streams reported by the tracking system are completely independent, some are much more so than others. For instance, the distance between the red/green and blue/yellow points in Figure 3 (i.e., those on the same arm) can be modulated by flexing the wrist upwards or downwards without a drastic change in the distances, angles, and centroids of points between the two hands. The angle of that same line is also relatively inde-

pendent. When rotating the forearm to change its angle, the upper point near the knuckle stays in roughly the same position, and the other arm can remain completely still.

Likely by-products of the movements described above are moderate changes in the lengths and angles of lines between the two arms. As mapping presets typically involve parameter assignments for all available control streams, this means that changes in even the most independent streams can cause audible effects not related to the primary intention of a particular movement. Though it is possible to try to avoid these side-effects by keeping the arms parallel, the multi-dimensional effects that accompany primary movements introduce a useful level of complexity to the system that can be understood and exploited with practice.

4.2. factorOracle

The factor oracle is an efficient automaton capable of producing at least all of the substrings of a particular input string (it may also find substrings that don't occur in the input). For our purposes, a string is a contiguous sequence of musical data—pitches, rhythms, or notes (combinations of pitch and rhythm), and a substring is any contiguous sequence of data within a parent string that is less than or equal to the length of the parent; we use the terms "sequence" and "subsequence" interchangeably with "string" and "substring" throughout, retaining the mathematical definitions of the latter two rather than the former.

Allauzen, Crochemore, and Raffinot describe an "online" version of the factor oracle construction algorithm, in which the automaton can be built incrementally as elements are added to the input sequence [1]. Elements of the input string become transitions between states in the automaton. Each state is followed by a transition to the next, and the following transition σ, taken from input sequence α, is related to the originating state β by:

$$\sigma = \alpha[\beta_n] \tag{1}$$

States are also connected to non-adjacent states by additional forward transitions that appear when the origin state for the transition ends a substring with a suffix identical to the suffix of a substring ended by the state to which the transition points. Backwards links, called suffix links, trace the path of a recursive supply function used to determine whether more forward transitions will be added when a new element is appended to the input sequence [8].

The algorithm for adding a new element, or "letter" to an existing automaton is shown in Algorithm 1. To add a new transition σ, we create a new state $(m+1)$ at the end of the oracle and add the transition from the previous state (m) to the new state. Then, we follow the previous state's suffix link to back to an earlier state k (this action describes the supply function $S_p()$ shown in Algorithm 1 below). If there is no transition from the earlier state to the new state by σ, we add it. We continue to follow the suffix links back and add transitions by σ when needed, until we reach state -1 or a state k_{final} that has a transition by σ to some other state. In the former case, we add a suffix link from $m+1$ to the $0th$ state, and in the latter we add a suffix link from $m+1$ to the state pointed to by σ from k_{final}. Algorithm 2 shows how to use this method to construct a factor oracle from scratch for any sequence of elements.

Algorithm 1 Function addLetter(Oracle($p = p_1\ p_2\ ...\ p_m$), σ) [1]

```
Create a new state m + 1
Create a transition σ from m to m + 1
k ← S_p(m)
while k > −1 and there is no transition from k by σ do
    Create a new transition from k to m + 1 by σ
    k ← S_p(k)
end while
if k == −1 then
    s ← 0
else
    s ← the state reached by transition σ from k
end if
S_pσ(m + 1) ← s
return Oracle(p = p_1 p_2 ... p_m σ)
```

Algorithm 2 Function buildOracle($p = p_1\ p_2\ ...\ p_m$) [1]

```
Create Oracle(ε) with a single state 0
S_ε(0) ← −1
for i ← 1 to m do
    Oracle(p = p_1 p_2 ... p_i) ← addLetter(Oracle(p = p_1
    p_2 ... p_{i−1}), p_i)
end for
```

Factor oracle automata allow us to switch seamlessly between analyzing and generating strings. We can build variants that sound more or less like an input sequence by controlling how we traverse the automaton: the greater the number of suffix links followed in succession before switching to forward transitions, the greater the dissimilarity between the output sequence and the original. Divergence between input and output strings also increases with an increase in the frequency with which we interrupt factor generation to follow a suffix link. In other words, the greater the ratio of suffix links to forward links chosen when traversing the oracle, the less the output sequence will correlate with the input sequence. This ratio can be roughly construed as the probability of congruence between input and output [2], [3].

We choose the factor oracle over algorithms for Variable Markov Modeling (VMM) such as Incremental Parsing (IP) or Probabilistic Suffix Tree (PST), because "fac-

tor oracles are functionally close to suffix trees [which describe the complete set of substrings within a string], but with fewer nodes. In comparison to IP and PST trees that discard substrings, factor oracles are preferred because they can be built quickly and, like the suffix tree, they encode all possible substrings [2]." Furthermore, both IP and PST methods need to "walk the tree from root to the node bearing the best (longest) suffix match;" with a suffix automaton, "the current state automatically models the best suffix, so there is no cost in searching it [3]."

The factorOracle object for Max has a constructor input for adding new elements to the automaton, an input for setting the output string length and an input for probability. As the probability increases, the oracle produces longer substring matches on the stored sequence built from values sent to the constructor input.

In the *Duoquadragintapus*, pitch and rhythm values generated by the performer feed the constructor inputs of parallel factorOracle objects. In *Cross-talk*, a single factor oracle automaton is fed with a cross-alphabet of pitch and rhythm values. In order to maintain a reasonable alphabet size, raw pitch values, which enter the system in cents, are downsampled to a target set of tempered pitch-classes, and raw rhythmic values are rescaled and quantized to a target set of time intervals. Each "letter" added to the automaton is associated with its original registration and amplitude, but these parameters are not used for pattern matching—they are only used for rendering the data to an output device.

5. CONCLUSION

Cross-talk is an environment for improvisation incorporating two performers playing gesturally extended instruments, and a virtual collaborator. All three entities share a parameter space: infrared motion-tracking information from the performers' hyperinstruments can be used as control data to affect transformations on the audio output produced by either or both instrumentalists; the software improviser builds material out of pitch and rhythm data from one performer, converging on or diverging from the input material based on probability values generated by the movements of either instrumentalist. There is a certain amount of unpredictability built into the system, which pushes the improvisational context; the software improviser may at any point begin generating data unrelated to the performance, and the effects of one instrumentalist's gestures on the other's output are often undefined.

The performance system incorporates previous work, including DILib for infrared motion tracking and the factorOracle Max external for machine improvisation. These software components were successfully integrated with gesturally extended instruments to exploit standard performance idioms, extend new resources to the performers, and provide a novel context in which to create music.

6. REFERENCES

[1] C. Allauzen, M. Crochemore, and M. Raffinot, "Factor oracle: A new structure for pattern matching," *Lecture Notes in Computer Science*, vol. 1725, pp. 295–310, 1999.

[2] G. Assayag, G. Bloch, M. Chemellier, A. Cont, and S. Dubnov, "OMax brothers: a dynamic topology of agents for improvisation learning," in *Workshop on Audio and Music Computing for Multimedia, ACM Multimedia*, Santa Barbara, USA, 2006.

[3] G. Assayag and S. Dubnov, "Using factor oracles for machine improvisation," *Soft Computing*, vol. 8, pp. 1–7, 2004.

[4] W. Brent, "DILib: Control data parsing for digital musical instrument design," in *Proceedings of the 4th International Pure Data Convention*, Huddersfield, England, 2011, pp. 176–180.

[5] S. Crab. (2012, January) Thaddeus Cahill's Dynamophone/Telharmonium. [Online]. Available: http://120years.net/machines/telharmonium/index.html

[6] E. Schwartz and D. Godfrey, *Music Since 1945*. Wadsworth, 1993.

[7] J. Williston. (2012, January) Thaddeus Cahill's Teleharmonium. [Online]. Available: http://www.synthmuseum.com/magazine/0102jw.html

[8] A. Wilson, "Automatic improvisation: A study in human/machine collaboration," Ph.D. dissertation, University of California, San Diego, 2009.