REAL-TIME SCRUBBING AND TRANSCRIPTION OF SCORE MATERIALS USING SCORESCRUB

Matthew C Lane

▶ **To cite this version:**

Matthew C Lane. REAL-TIME SCRUBBING AND TRANSCRIPTION OF SCORE MATERIALS USING SCORESCRUB. Journées d'informatique musicale, May 2017, Paris, France. ⟨hal-03353801⟩

HAL Id: hal-03353801

https://hal.science/hal-03353801v1

Submitted on 24 Sep 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

REAL-TIME SCRUBBING AND TRANSCRIPTION OF SCORE MATERIALS USING *SCORESCRUB*

Matthew C. Lane
Université de Montréal
matthew.lane@umontreal.ca

ABSTRACT

The author presents his software, ScoreScrub, a computer-assisted composition system used for improvisatory real-time scrubbing of existing notated musical material and its transcription. Employing gestures received by a digital stylus and graphics tablet, the user can improvise on fragments of existing music, backwards or forwards, and at any speed by "writing" on the original score. The improvised passages are simulated in real-time through MIDI, then transcribed and quantified to adapt them to performance by musicians. The software features pitch, time, and real-time orchestration controls, all of which can be controlled by the vertical or pressure components of the stylus movement, or by a MIDI-controller. The article examines the compositional problem and inspiration that catalyzed the software's creation, including some existing technology and specific difficulties in the traditional compositional process. Further explanation is devoted to its specific goals and features, practical concerns in its development, and finally, its use in a piece that was premiered in the fall of 2016, entitled "Lullabies for Boys Who Will Not Sleep Anyways."

1. IMPROVISATORY COMPOSITION

There has long been a partial divide between the act of improvisation and the act of composition in instrumental music. While both are generative musical processes, they tend to be perceived as having different goals and different strengths. Kahneman's differentiation of thinking systems 1 and 2 provides a simplistic means of regrouping these activities, with system 1 encompassing the impulsive, the emotional, and the automatic (improvisation),[1] and system 2 encompassing the analytical, the weighted, the structured (composition, although not without overlap). [14] Indeed, Larson's "traditional" definition of improvisation perhaps fits best with my own experience as a composer and performer:

"Improvisation is traditionally regarded as a process in which performers, with their voice or instrument, in 'real time,' use luck or skill to respond to or incorporate mistakes; the improvisation grows out of innovation, exploits freedom, and relies on talent in an instantaneous process that involves emotional invention and intuitive impulse to create simple, direct expressions." [16]

Several of these attributes, including "emotional invention and intuitive impulse," as well as the response to or incorporation of "mistakes" make the possibility of composing in "real-time" appealing, even if the result is not immediately available to an audience, as with an improvisation. The required "talent", however, that an experienced jazz musician has, may place the possibility of improvising or composing in real-time beyond the aptitudes of many composers, not to mention the challenge of notating an improvisation after the fact. This difficulty is taken further when multiple planes of tone are involved, often meaning multiple instruments, or at least multiple independent lines.[2]

The need for immediate expression is not limited to music. My personal experience from a young age demonstrated the need to find means of expressing myself faster than the speed of writing, and psychologists recommended typing lessons for me at an early age (in the early '90s, before such things were widely taught in schools). Having followed me my whole life, computers seemed a useful way of surpassing the limitations of my natural speed of transcription, whether with a pencil or with notation software. Since my goal was to develop a tool for composition, and not performance, it fell into the realm of computer-assisted composition. This will be a recurrent theme in this article: while the tool shares features with a digital instrument, it is intended as one step in the compositional process, and not for the performance of music. The term "improvisatory composition" has been helpful in explaining this system.

2. TRACING THE ROOTS OF *SCORESCRUB*

2.1. Scrubbing

The aforementioned challenge has a multitude of possible solutions, so some explanation is useful as to why scrubbing with a digital stylus and graphics pad

[1] This is not intended to disparage improvisation, or to depreciate the high degree of learned instrumental skill, theory, and analysis involved in improvisation. The meaning is to suggest that these characteristics:

[2] "Planes of tone," [3] a concept from Alan Belkin's treatise on orchestration (and reportedly from D.F. Tovey) refers to any instruments, voices within an instrument, or group of instruments sharing a "rhythmic outline."

was eventually chosen as the means to improvisatory composition.

"Scrubbing", for our purposes, represents the ability to move backwards and forwards in a sound (or sometimes video) freely in real-time, and at varied speeds. It also allows jumping to different points in an audio or video stream. Figure 1 shows how scrubbing could work for notated music, with arrows representing the movement through the source material.

The idea of scrubbing is not new: the concept derives from "scrubbing" tape across the magnetic head [21], and has been adopted by DJ's, most waveform editors, and most recently, cellphones. The notation software *Finale* even contains a similar little-known feature, allowing users to scrub across segments of their score either forwards or in reverse, and at different speeds. [12]

A number of more powerful research-based scrubbing systems also exist, although primarily for sound. Some, such as DiMaß, scrub audio with high fidelity at the original pitch [17]. Some composers have also developed similar structures for electronic music improvisation, such as Doug Van Nort's *greis* system [21], or the Bricktable system by Hochenbaum and Vallis [11]. Some research has even explored the addition of haptic feedback to audio scrubbing devices [7].

Many other creative interfaces resembling scrubbing tools exist for both sound and notation, although most are generative instead of using existing musical material, and a large proportion of them are devoted to performance as opposed to CAC. Devices like ReacTable [13] or Small Fish Tale [9] share some of the gestural control elements of a scrubbing mechanism, but don't work with existing material as much as producing new material. They also work primarily with sound, although MIDI outputs mean they could be easily be adapted to produce scores.

IMPI enables a conductor to draw a score in real-time for performers [9], but is still primarily for generating new material rather than scrubbing existing material. It's gesture paradigm and capacity for producing scores, however, resemble what one might do while scrubbing, and some of it's mapping characteristics are similar to the concept of plugins explored in section 3.2.

CAC-devices exist that allow processes similar to scrubbing, like the drawing of musical lines in Moz'Lib [10]: here, like with scrubbing, the temporal aspect can be controlled, and one has visual and audio feedback, but again, it is ultimately concerned with the production of new material than the reinterpretation of existing material. Musink [19] does work with existing material, allowing one to augment a score digitally, but does not deal with the movement back and forth in time like scrubbing does. And OpenMusic programs like those used in some of Philippe Leroux's works (see section 2.2 for more on this) are again generating new material rather than working with existing material.

The idea of scrubbing scores was attractive because it would allow the user to work with existing material, both creating unity in a final result, and allowing the composer/improviser to focus on the movement of the material instead of the creation of material itself. Little would be required in terms of musical instrument skill, allowing what Wessel and Wright referred to as "low entry fee with no ceiling on virtuosity". [24] Much like jazz players might use a rhythmic motive and a scale as a means of improvising, the user could do the same without the instrumental expertise.

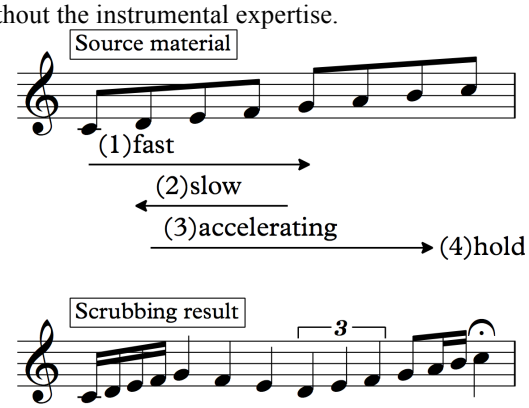

Figure 1 An example of how notation scrubbing might look.

Improvising based on existing material provides other advantages. One can improvise with many layered and independent lines (imagine improvising from sections of a fugue or a complex orchestral work). One is not limited to the "one gesture to one acoustic event" model of traditional instruments. [24] Also since the process would not ultimately be "generative", as is often the case in electronic improvising instruments, [24] but rather "transformative", certain kinks could be easily worked out in advance. Impossible notes may be excluded from the source material, as may be unbalanced textures, for example. Other idiomatic instrumental issues can be avoided before the improvising even begins.

Most modern devices allowing scrubbing also incorporate a visual interface allowing you to view a waveform, a concept that translated well since western instrumental composers are generally used to composing in a visual domain, with an x-axis that more or less represents time.

2.2. Digital stylus and graphics pad

To scrub, a graduated axis of some sort is required. It could be a slider, an arm movement, even a pressure gradient. Any number of input devices could be used to control such scrubbing software, with the two main categories being those that exploit existing motor control abilities, and those deliberately avoiding previously learned motor-control abilities. [23] While there are advantages to working in a completely new paradigm, I sought to capitalize on the association many composers already have between a score and a pen. Such a stylus system, like that in Figure 2, could be directly linked to the score sample displayed on the screen, exploiting an existing link between score-

creation and pen use, in the same way computer users automatically link between a mouse movement and its screen placement. The same x-axis that represents time in a score could be translated to the graphics pad's x-axis.

Figure 2 Wacom digital stylus and graphics pad

Finally, modern graphics tablets can sense a pressure gradient, allowing a third axis of control from a single point. It's one of the most concentrated means possible to send three axes of data with a single hand, while sitting comfortably at a desk (not including, for example, the use of motion sensors).[3]

Using tablets as a generative or transformative source for instrumental music has also been well-documented for instrumental works like Philippe Leroux's *Apocalypsis*. [22] More recently, Leroux's piece *Quid sit musicus* even used real-time features in OpenMusic to produce instantaneous visual and audio feedback of pen gestures [9], similar in some respects to features that would be desirable for scrubbing software.

2.3. Personal influences and ideas

Several factors specific to my process contributed to the planning for this software. Firstly, it is very much a real-time extension of previous research-creation I did using break-point-functions to generate progressions from existing musical motives. [15] Many of the components of this software are near-copies of similar processes developed earlier in OpenMusic.

Since my work has so long lived in a space between the electronic and instrumental worlds, I've long sought to transfer or "transpose" concepts from electronics. My composition tends to be instrumental, or electronic within an instrumental paradigm, but my history with computers-based processes is extensive.

One of these transpositions is "effects", similar to plugins: a hallmark of any DAW is the processes that can be applied in real-time to a sound source. ScoreScrub explores some of the possibilities for how this might work in an instrumental context. While certain notation software already allows plugins, they are generally static. A real-time application of, for example, pitch shifting based on pen pressure is possible in this system.

Finally, elements of my style dictated some of the direction taken by the software. The use of scrubbing was ultimately conducive to techniques like cross-cutting (think Stravinsky, Michael Finnissey [1], or Elliot Carter [6]) due to the ability to jump around freely in a score sample. The development of short, motivic rhythmic gestures are also well served by this system, due to the ease of repeating or slightly altering short fragments using impulsive, quick gestures.

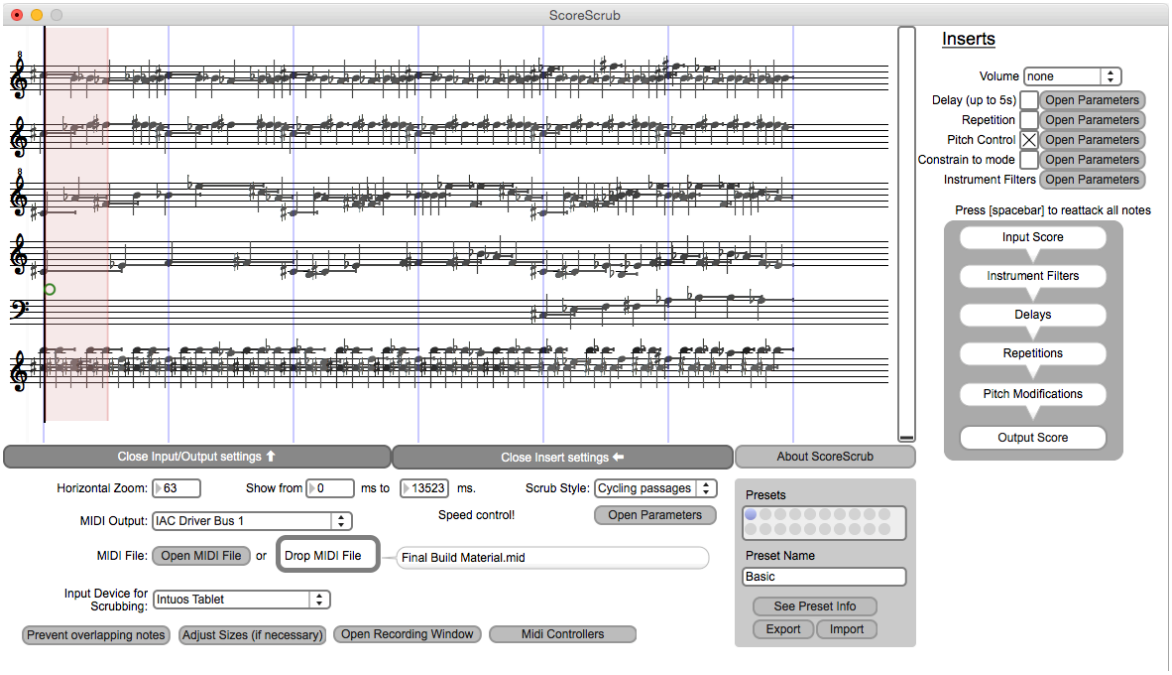

Figure 3 The main window of ScoreScrub

[3] Several composers, for example have made use of the Kinect software for XBox 360 in order to generate instrumental scores [4]

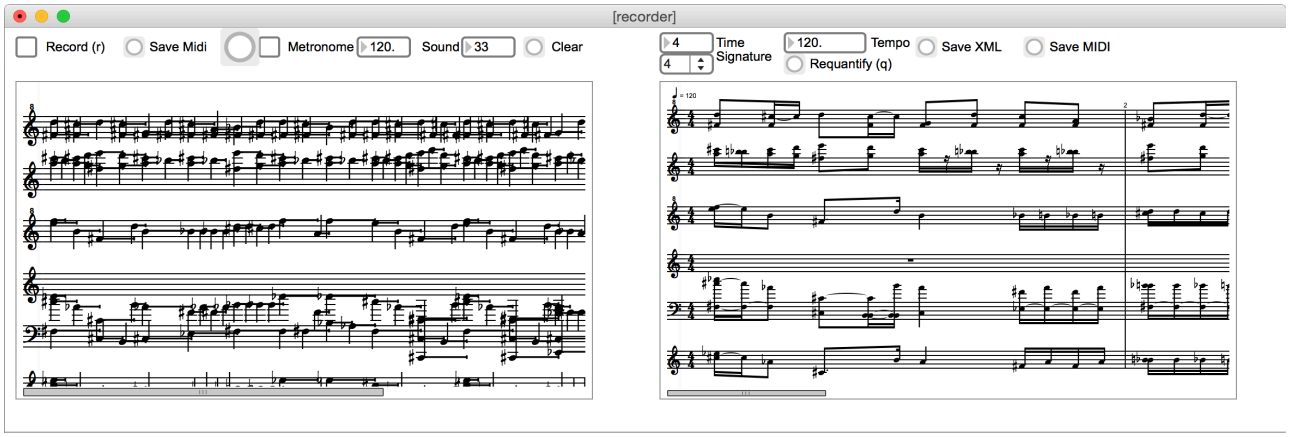

Figure 4 Recording and quantification window

3. SCORESCRUB FEATURES

ScoreScrub was developed in MaxMSP using *bach automated composer's helper* for reasons that will be explained in Section 4. First, however, it is worth examining which functions[4] were desired by the author, and how they were eventually implemented in the program. These functions are broken down into three categories of importance, representing the author's approach to designing the software:

- *Primary functions* are those that define the software, and serve as the basis for what it was intended to do.
- *Secondary functions* provide additional capabilities to the primary functions, significantly increasing the possibilities for what the software can do.
- *Tertiary functions* enhance the software, streamlining tasks, but do not fundamentally change the capacities of the program.

These are explained from the standpoint of a composer, but for those interested in the program's architecture, a link for the source code may be found at the end of the article.

3.1. Primary functions

While not intended as an electronic performance instrument, but rather as a tool for CAC, I wanted the program to incorporate certain characteristics of the former to better respond to the impulsive gestures of the composer. The first primary function required by the software, in order to function with some of the capacities of an "instrument" for computer-assisted composition, was immediate auditory feedback of what is being scrubbed, obviously including feedback of any effects applied the material. To make it useful as computer-assisted composition, it required a practical transcription function. Finally, like an instrument, it should have a silencing feature, which was easily accomplished by detecting when the pen was lifted. [24] These combined factors serve to create a link between the physical motion and the auditory/visual feedback.

The scrubbing window simply looks like a score without rhythm markings, since rhythm is completely proportional. (Figure 3) A small green circle, referred to as the "cursor" (between staves 4 and 5, to the left, in Figure 3), indicates the location of the stylus on the score. In this basic mode of functioning, dragging the pen left across the pad (and thus moving the cursor left across the screen) will essentially play the passage in reverse, and moving the pen right will play forwards. (see Figure 1 for clarification) Depressing the pen will attack the notes, and lifting it will release them. The pen position on the y-axis, or the pen pressure may be assigned to other secondary functions, such as volume or transposition.

In terms of quantification of the scrubbed score segments, "practical" was the key requirement. While many algorithms exist for quantification, if a truly performable musical work is sought, a balance must be reached between absolute perfection and playability. The quantification window of the software initially records the performance in timed notation, and the composer can subsequently test quantifications with varying tempos and time signatures to find the variant that suits the score best. (Figure 4)

3.2. Secondary functions

The most important secondary feature for my music was the possibility of dynamically cycling through portions of a passage. Like in a DAW, a certain passage can be cycled repeatedly, which suits much of the obsessive and juxtaposition-laden music I compose. But in the case of ScoreScrub, the cycled region can be moved dynamically. In cycling mode, as the stylus hovers over the pad, a red box appears showing the region to be cycled, with the pen's location designating the start point, and the pressure or the vertical axis designating the length. (Figure 3, the box is gray in the image and covers the leftmost portion of the score) Then

[4] The term *function* here should not be confused with the programming term, but comes from the value analysis definition as "the purpose that a product, project or process is expected to perform" [20].

when the pen is depressed, the software will begin cycling through (and playing and transcribing) the highlighted passage. If the pen's placement is adjusted while it plays, the cycling region will be adjusted when the next cycle starts, with the user receiving a preview of this location via the red box. Finally, the cycling stops when the pen is lifted.

This concept expands on one of the several ways scrubbing can work in the digital world. In some cases, scrubbing works through resampling or time-stretching, but in other cases it works through "skipping": playing a short fragment from one section at a time as the cursor moves. [17] Here this concept is simply expanded to a much larger time sample, more representative of instrumental music than pure sound. It could also be seen as a form of macro-granulation.

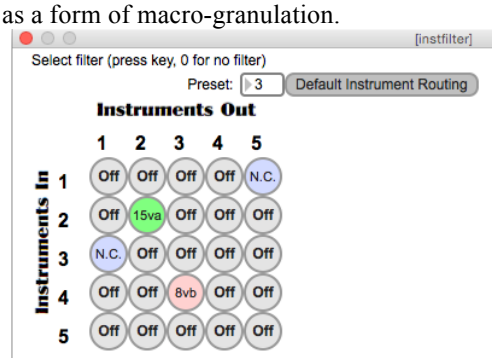

Figure 5 Orchestration/instrument routing window

The next most significant feature was some limited, but useful, dynamic orchestration. A matrix is used to route "input instruments" (the staves on the original score material) to "output instruments" (the staves on the final transcription, and the simulated instruments), with certain specific possible transpositions. (Figure 5) For example, the flute from the original score segment could be sent out for playback and transcription by the clarinet, but transposed down an octave. Or perhaps some instruments should cut out completely during some segments. A number of presets, controlled by the number keys, trigger dynamic switching between different "orchestrations", allowing the juxtaposition of different textures or levels of depth to the music.

Effects, similar to plugins, as mentioned above, play a significant part in the program. Much like the kind of processes that can be accomplished in software like OpenMusic, certain effects were created to allow altering the pitch or rhythm in different ways in real time. In most cases, the user can control which instruments are affected, and the degree of the effect can be mapped to the y or z (pressure) axes of the tablet, or to a MIDI-controller. Table 1 shows the effects currently functional, how they change the scrubbed music (Main Function), and which parameters are available to the user for more detailed control.

Effect	Main Function	Parameters and Control
Velocity	The output notes' velocities (volumes) are changed.	The velocity is controlled by a stylus-axis or MIDI-controller.
Delay	Delays each instruments by different millisecond amounts.	Individual delays are supplied for each instrument. Lengths of delays can be influenced by a coefficient based on a stylus-axis[5] or MIDI-control.
Repetition	Causes held notes to be repeated after a specific millisecond amount.	Repetition time can be influenced in real-time by a stylus-axis or MIDI-control. Can be bypassed for some instruments.
Semitonal transposition	Transposes the output by a variable number of semitones.	The transposition is controlled by a stylus-axis or MIDI-controller. The upper/lower transposition bounds and a transposition curve function can be set. Like all pitch-related functions, the effect may bypass certain instruments, and it may be set to only change between cycles in cycling mode.
Specific interval transposition	Transposes the output by one of a number of specific intervals.	A stylus-axis or MIDI-controller determines which transposition interval is used. The number of transposition intervals may be changed and the intervals themselves may be adjusted (ie. 3,7, and 12 semitones up).
Frequency shifting	Frequency-shifts the output.	A stylus-axis or MIDI-controller changes the modulator frequency. The upper and lower modulator frequency bounds, as well as a curve function, can be set.
Constrain to mode	All output (after other processes) is constrained to a specified mode.	The mode or pitch-field can be set by the user.

Table 1 Effects in ScoreScrub

3.3. Tertiary functions

Other features were slowly added to encourage a more fluid creative process. The most obvious were features for easy importation from existing scores via MIDI or XML, and the subsequent exporting of a transcribed result in similarly portable formats.

As the program became more sophisticated, the next feature was a preset mechanism for the retrieval of existing setups, including the source files used, the layout, the effects mappings, and orchestration presets.

For more rhythmic music, and for better precision, adjustable gridlines and grid-snapping were both added, allowing the maintenance of rhythmic integrity from the original material.

[5] A stylus-axis refers to either the y or z (pressure) axes on the graphics tablet.

3.4. Further features

Following is a brief list of further features that were added to further facilitate the computer-assisted composition process:

- MIDI-device mapping: this feature allows external MIDI-controllers the possibility of controlling various parameters. (Figure 6)
- Prevent overlapping notes: scrubbing scores often works best without overlapping notes. (when scrubbing backwards, a note's release becomes its attack)
- Horizontal zoom and begin/end times for the score window.
- Two types of mouse scrubbing, in case a stylus and tablet is unavailable.
- A metronome, linked with the temporal grid being used.
- Buttons to both "freeze" the music in cycling mode, and to re-attack all held notes.

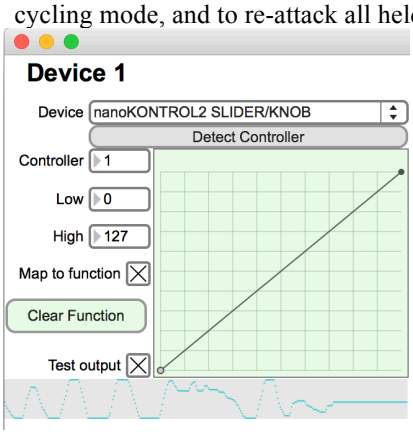

Figure 6 MIDI-controller interface (one of 3 possible device inputs)

4. PRACTICAL CHOICES IN DEVELOPING SCORESCRUB

4.1. MaxMSP

Due to the real-time nature of these goals and the desired graphics interface, MaxMSP was a logical starting point for the software. Low latency is crucial to any type of improvisatory instrument [24], and while software like OpenMusic has recently expanded into some real-time processing,[6] there are still issues better solved in audio-processing software like Max or PureData.

MaxMSP is designed with some flexibility in terms of input and output, working well with MIDI controllers and external midi devices (for simulation). The relative ease of programming and reliability of Max made it a fairly straightforward choice, but enhancing that were two existing Max programs that were essential to the software.

The first was the *s2m.wacom* patch, which interfaces with the Wacom/Intuos stylus and tablet that is currently the most easily available in North America.[7] The patch, written by Charles Gondre from CNRS-LMA, retrieves all the necessary information from the tablet at regular intervals, including x, y, and z values, and any buttons pressed, saving significant amounts of reprogramming.

4.2. bach automated composer's helper

MaxMSP, however, lacks a built-in notation interface, of the kind that might be found in OpenMusic, engraving software, or even most DAWs. *bach automated composer's helper*, by Andrea Agostini and Daniele Ghisi, is a powerful Max-based engine designed with computer-assisted composition in mind, and it enables full notation integration in MaxMSP. [1] *bach* uses a similar structure to LISP, the base language of OpenMusic, and all score materials can be controlled and retrieved via *Lisp-like-linked-lists* (llls).

This made the transition from OpenMusic especially easy, and allowed me to incorporate some of the same processes I had used previously without significant reprogramming. They could now, however, be applied in real-time, with a versatile visual interface. The notation objects in *bach* could function as a high-level graphics user interface to control the real-time MIDI-simulation, and to send the necessary material for transcription as a new score.[8]

4.3. Compromises and drawbacks

While the software has thus far achieved most of its intended goals, there remain some limitations, and they merit a short discussion:

4.3.1. Processing power

The software requires significant processing power, due to the extensive visual processes, sequencing effects, and audio feedback required to function in real-time. It is not ready yet for low-powered laptops, and certainly not tablets.

While the latency occasionally passes the 7-10 ms suggested by Wessel and Wright for digital instruments, and variations in latency can surpass the suggested 1 ms [24], internal time-code tags have helped reduce the effects of this in the final transcription of an improvisation.

[6] This refers to the "reactive" features that have been added to OpenMusic. [5]

[7] A simple glance at Amazon's best sellers list, and the inventory at most Canadian computer stores, makes it clear that for the moment, this is the most easily available of such tablets.

[8] In the final program architecture, it was not efficient to use the *bach* object as the actual user interface. Instead, an invisible layer (a Max *lcd* object) above the *bach.roll* object interacts with the input device, giving the impression that the user is directly influencing the *bach* object.

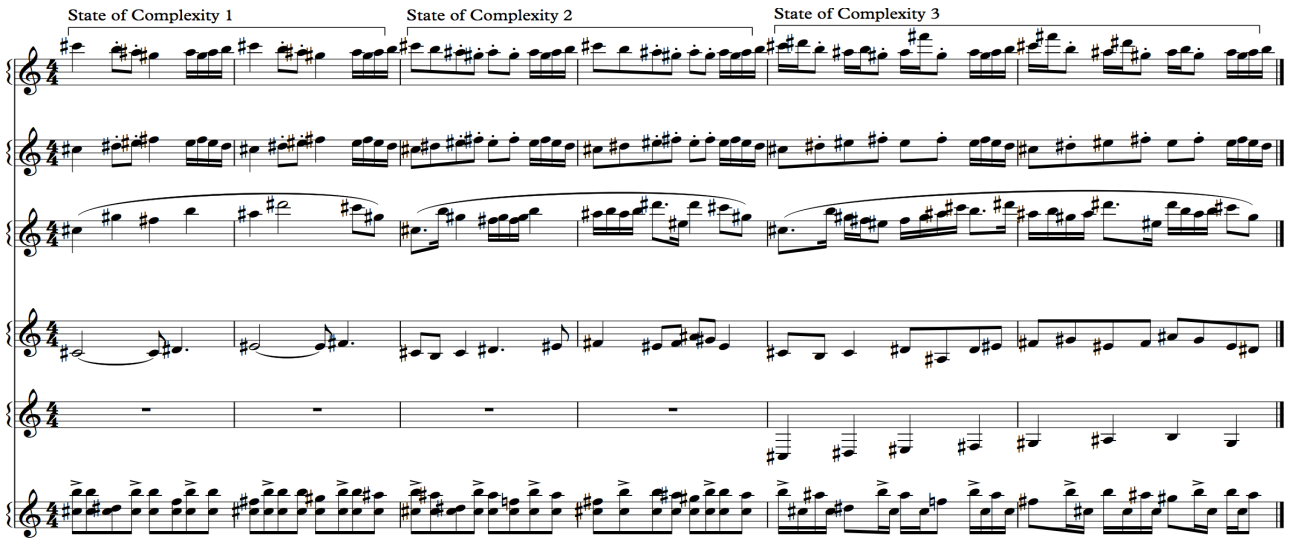

Figure 7 *Lullabies...* source material for ScoreScrub

Even with time-codes, however, quantification over long improvisations can become imperfect. While, in the author's music, the software was only intended for short-medium length sections (ie. up to 2 minutes), this may eventually pose a problem for longer sequences.

4.3.2. Digital stylus and graphics tablet required

The software's greatest asset, the pen-based interface, also requires an additional hardware investment. And due to the adjustable scaling of the score interface, the difficulty in pinpointing a precise musical gesture can change drastically between setups.[9] While this provides significant flexibility, it can also mean relearning a new "instrument" with each piece, similar to moving between a violin and a viola, for example.

4.3.3. Sound libraries required

The software's output is in MIDI, passing through the computer's IAC busses (or directly to a 3rd party MIDI-bus, as the case may be). Consequently, quality feedback from the score section being sampled requires quality sounds. Many instrumental composers, however, regularly work with such sound libraries (in conjunction with their notation software or otherwise), so this doesn't necessarily present a further investment. In a worst-case scenario, the computer's default MIDI program may still provide a decent enough approximation for compositional improvisation.

4.3.4. Counterpoint

The most significant musical consideration, and one that will ultimately restrict the software's use to only certain compositional practices, is its inability to handle true counterpoint. This is due to the singular x-axis control for time, meaning all music moves implacably together in most cases.

Incorporated effects such as varied delays for different instruments can create a sense of counterpoint, but the voices are still ultimate interdependent.

For one piece in which I made use of ScoreScrub, *Short Pieces on Falling: Waves*, contrapuntal source material was used for scrubbing with some success. Each time a motive in one voice returns, however, the associated counterpoint in other voices consequently returns, and the ideas become inextricably linked for the listener. They thus shed some of their independence. There is room for more contrapuntal adaptability in the future, however, with the increasing availability of multi-touch track pads and graphics tablets.

5. *LULLABIES FOR BOYS WHO WILL NOT SLEEP ANYWAYS*

The fourth movement of *Lullabies for Boys Who Will Not Sleep Anyways* made extensive use of ScoreScrub in its final section. The work is composed for piano, toy piano, and fixed audio. The fixed audio part consists, amongst other sounds, of four additional artificial instruments created from sampled toy instruments, in some cases processed with velocity- or pitch-dependent effects. In all, this allowed for a virtual ensemble of six instruments.

The source material inputted into ScoreScrub (Figure 7) was a set of subjects and countersubjects: 6 superposable melodies, each with three states of complexity (from simple and transparent, to complex and busy).

5.1. ScoreScrub setup

For this piece, the cycling mode was used, with a snapping-grid, to bring out the strongly rhythmic and motivic nature of the source material. In cycling mode, music always plays forward locally, but the sections

[9] Different zoom factors create varying results of Fitt's law. See [23]

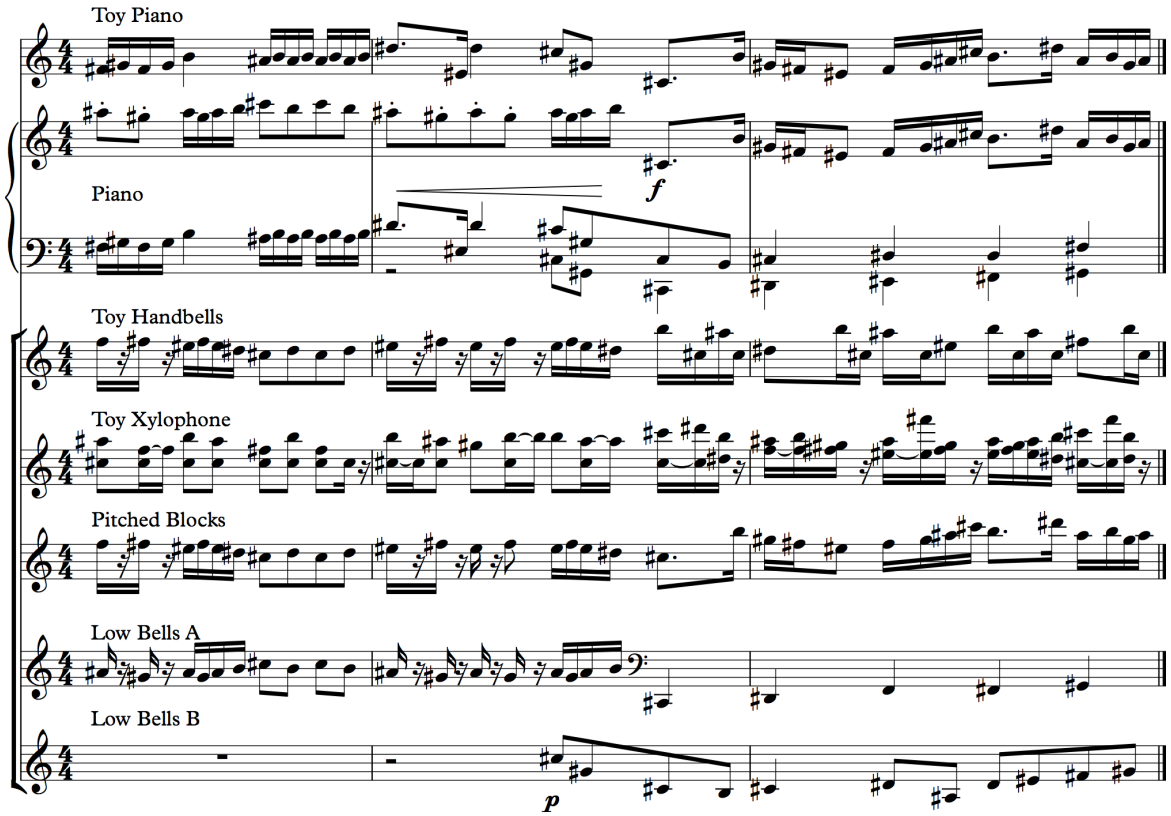

Figure 8 Material generated in ScoreScrub for *Lullabies for Boys Who Will Not Sleep Anyways*

being played can jump around and their respective lengths can change based on pen movement. The x-axis of the pen therefore controls the starting point of the segments being played, and the y-axis controls the length of each segment played (considered as a sort of macro-granulation for notation, the y axis controls the sampling window).

The z-axis, or pressure, in this case controlled a frequency shifting effect, which was applied only to three of the instruments (in this case, the synthetic instruments, and not the human-played instruments). With some spectral effects applied to the instruments,[10] it allowed the creation of a cloud of quasi-instrumental sound that floated around in pitch during repetitive bits in the instrumental part. The frequency shifting of each pitch was from 0-300 Hz, depending on the pressure (0-1). The formula could be written as follows: (where F_1 is the initial note frequency, Z is the pressure axis from 0 to 1, and F_2 is the final frequency):

$$F_2 = F_1 + (300Hz \times Z)$$

Finally, a set of orchestration presets was established to allow dynamic instrument starting, stopping, transposition, and routing. In the preset shown (Figure 5), instrument 1's material is routed to instrument 5 at the same pitch, instrument 2's material is routed back to instrument 2 two octaves higher, and instrument 4's material is routed to instrument 3 an octave lower. Finally, instrument 3's material is routed to instrument 1 at pitch.

[10] This was done using Michael Norris' *Soundmagic Spectral* plugins, a freeware suite of plugins capable of real-time spectral processing.

5.2. Results and further editing

The scrubbing process was undertaken several times, each time with nearly complete freedom of improvisation, but the awareness that, like a recording sessions, more takes were easily possible. The only structure planned was several alternations between high- and low-pitched valenced textures, and an eventual progression towards obsession.

In the final performance score, only the piano and toy piano parts are shown, but Figure 8 shows an approximation of what all the instruments do in one passage. Note the obsessive returns of motives, as well as the change of orchestration preset partway through the second measure.

Small changes were subsequently made to the score to achieve the result sought in the improvisation, but for which the software did not provide sufficient flexibility. For example, in some cases, instead of the orchestration being changed abruptly, it was altered to transition more gradually. In some cases, on re-listening, a beat was added or removed, or a pedal tone was added in an instrument.

It is, after all, computer-assisted composition, and not simply computer composition. The final say in my instrumental music say should always belong to me, and should be able to transcend the limits of whatever software is used, only constrained by the limitations of the performers and the imagination.

6. FINAL THOUGHTS

ScoreScrub has thus-far proved to be a intuitive, responsive and even musical interface in five different works, providing the tools needed to improvise segments of compositions based on composed source material. The results have proven to be playable, interesting, and requiring little post-export editing.

For the author, few features are lacking that would make the software more intuitive without making it significantly more draining on the system, or more complex to set up and use. Out-of-the-box usability is crucial in the creative process, and one of the most important functions the software provides is a means to hopping over the analytical programming process directly to the intuitive improvising process.

The software is now ready for open beta-testing,[11] and several suggestions are already being considered for the next version, including micro-tonality, dynamic pitch adjustment (glissandi, for example), and the incorporation of electronics parts.

Like any improvisational or creative tool, the proof is in the composer's sense that the device can capture their intuitive expression. So far, the signs are positive, but it will take more composers to truly judge.

7. REFERENCES

[1] Agostini, A. and Ghisi, D. "A Max Library for Musical Notation and Computer-Aided Composition", Computer Music Journal, Volume 39, No. 2, p. 11–27, 2015.

[2] Beirens, M. "Archaeology of the Self: Michael Finnissy's 'Folklore'", *Tempo*, Cambridge University Press, Vol. 57, No. 223 (Jan.), p. 46-56, 2003.

[3] Belkin, A. "Basic Notions, Part 2", *Orchestration*. http://alanbelkinmusic.com/site/en/index.php/orchestration/, 2001.

[4] Berg, T. et al. "Interactive Music: Human Motion Initiated Music Generation Using Skeletal Tracking By Kinect", Proceedings of the Conference for the Society Electro-Acoustic Music in the USA, https://vision.cs.unc.edu/home/publications/kinect_music.pdf, 2002.

[5] Bresson, J. *Reactive Visual Programs in OpenMusic*. [Research Report] IRCAM, https://hal.archives-ouvertes.fr/hal-01142078/document, 2014.

[6] Bye, A. "Carter in Context", *Tempo*, Cambridge University Press, No. 176 (Mar.), p. 61-62, 1991.

[7] Chu, L. *Haptic Interactions for Audio Navigation*. Ph.D. Dissertation. Stanford University, Stanford, CA, USA, 2004. Chris Chafe, advisor.

[8] Deliège, C. "Indétermination et improvisation", *International Review of the Aesthetics and Sociology of Music*, Croatian Musicological Society, Vol. 2, No. 2 (Dec.), p. 155-191, 1971.

[9] Garcia, J. et al. "pOM: Linking Pen Gestures to Computer-Aided Composition Processes", *40th International Computer Music Conference (ICMC) joint with the 11th Sound & Music Computing conference (SMC)*, Sep 2014, Athens, Greece. 2014.

[10] Ghisi, D. et al. "Extending bach: A Family of Libraries for Real-time Computer-assisted Composition in Max", *Journal of New Music Research*, 46:1, p. 34-53, 2017

[11] Hochenbaum, J. et al. "Bricktable: A musical tangible multi-touch interface." *Proceedings of the Berlin Open*, 2009.

[12] Johnson, T. "Scrubbing Your Music?", *Finale Blog*, http://www.finalemusic.com/blog/scrubbing-your-music/, 2010.

[13] Jordà, S. et al. "The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces." *Proceedings of the 1st international conference on Tangible and embedded interaction*. ACM, 2007.

[14] Kahneman, D. *Thinking Fast and Slow*. Anchor Canada, 2013.

[15] Lane, M. "Programming modular progressions in OpenMusic" *The OM Composer's Book 3,* ed. Bresson, J. et al., IRCAM, Paris, p. 55-75, 2016.

[16] Larson, S. "Composition versus Improvisation?" *Journal of Music Theory*, Duke University Press on behalf of the Yale University Department of Music, Vol. 49, No. 2 (Fall), p. 241-275, 2005.

[17] Lee, E. et al. "DiMaß: A Technique for Audio Scrubbing and Skimming using Direct Manipulation" *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*. ACM, 2006.

[18] Lee, E. et al. "Improving Interfaces for Navigating Continuous Audio Timelines", Media Computing Group at RWTH Aachen University. https://hci.rwth-aachen.de/materials/publications/lee2007b.pdf, 2007.

[19] Tsandilas, T. et al. "Musink: Composing Music through Augmented Drawing." *International conference on Human factors in computing systems*, Boston, United States. p. 819-828, 2009.

[20] Value Analysis Canada. "Value Analysis Definitions", *Value Analysis Canada*, http://www.valueanalysis.ca/vadefinitions.php?lang=en, 2015.

[21] Van Nort, D. "Multidimensional Scratching, Sound Shaping and Triple Point", *Leonardo Music Journal*, The MIT Press, Vol. 20, p. 17-18, 2010.

[22] Vassilandonakis, Y. and Leroux, P. "An Interview with Philippe Leroux", *Computer Music Journal*, The MIT Press, Vol. 32, No. 3 (Fall), p. 11-24, 2008.

[23] Wanderley, M. M. and Orio, N. "Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI", *Computer Music Journal*, The MIT Press, Vol. 26, No. 3 (Autumn), p. 62-76, 2002.

[24] Wessel, D. and Wright, M. "Problems and Prospects for Intimate Musical Control of Computers", *Computer Music Journal*, The MIT Press, Vol. 26, No. 3 (Autumn), p. 11-22, 2002.

[11] A beta version is available at www.matthewclane.com, along with source patches that reveal more of the program's internal architecture.