Implementation of an 8-Channel Real-Time Spontaneous-Input Time Expander/Compressor

Introduction:

The ability to time stretch and compress acoustical sounds without effecting their pitch has been an attractive compositional technique for many years. In his article *Discovering Inner Complexity: Time Shifting and Transposition with a Real-time Granulation Technique* Barry Truax describes, "as a sound is progressively stretched, one is less aware of its temporal envelope and more aware of its timbral character...which with natural sounds is amazingly complex and musically interesting" [Truax 94]. My own experience using the technique was with Richard Karpen's 1992 Csound module '*sndwarp*' which I used in the composition of *The Ghost Withi*n [Keyes 1998] for piano and tape. While the essential and recognizable timbre of piano samples used were preserved, the rich grainy quality and time varying textures that resulted were quite effective in that context. Many of the sounds were created by time stretching sampled piano sounds 10 fold, and then compressing them 10 fold to maximize the textures created (and not actually changing the length of the samples). I also used Bill Schottsaedet's *SND,* which uses a similar algorithm in the composition of *Li Jiang Etude No*. 1 [Keyes 1999]. As this was a "study" in Chinese instrumental timbres the technique again proved extremely useful in this context.

The '*SND*' program actually runs in "real time" on the SGI, meaning that once your sound was loaded you could hear the results in real time. However, being much more fascinated with the application of real-time DSP in live performance situations, I longed for a way in which the technique could be applied to samples coming directly from a microphone with no audible delay. This, I imagined, would go even further to combine the "complexity associated with studio composition with the spontaneity of live performance" [Truax 1988]. I also longed for a way to spatialize the output over many channels such that different grains could be spread over a much more vast sound field. Lastly, I wanted a Graphical User Interface (GUI) that would allow a musician to control it easily on stage. The MAX/MSP "patch" discussed below has so far accomplished this successfully, and I hope a brief discussion of problems addressed and features implemented will stimulate interest in its use and/or save time in some one else's implementation.

Brief Review of Granular Sampling Technique:

Others may have explained this in greater detail (see Roads 1978 and 1991) but in brief, the most common approach to time expansion/compression now used begins by taking slices of samples from a buffer and applying an amplitude window to them. Each 'slice' of samples is often referred to as a 'grain' and thus the processes referred to as 'Granular Sampling'. Grains are then overlapped so that one does not notice (as apparently) the amplitude changes that result from the windowing (though they usually are perceived at another level, and that is one of the attractions of the technique!). Thus the main parameters are the size of the grains (usually expressed in milliseconds) and the degree of overlapping with other grains. These grains are then all advanced through the buffer by a separate operator, in the case of figure 1, the large arrow.

Figure 1: Granular windows of a sampled sound.
a-Window size
b-Overlap

Each of the grains is read by a phasing operator, which continuously loops from the beginning of the grain to its end at a given frequency. The frequency of resultant pitch is thus determined by the frequency of the phasor. The length of the resultant sound, however, is controlled separately by the rate in which these grains advance through the buffer. Figure 2a shows a simplified version of such a granular engine in MAX/MSP. Here the phasor element is accomplished by a 'phasor~' object, shown there with a frequency of 20 hertz. As the phasor only counts from 0 to 1, this increment is then multiplied by 50 for a grain size of 50 milliseconds. This again controls the frequency and only the frequency. On the left is the object which advances the grains through the buffer, in this case a 'line~' object. Although this simplified engine will actually work, one will hear discontinuities (clicks) as the 'line~' operator moves the grains through the buffer because there is no windowing operation taking place. By applying a window element such as in figure 2b, and by advancing the line~ object only when the window is at zero this problem is easily overcome.

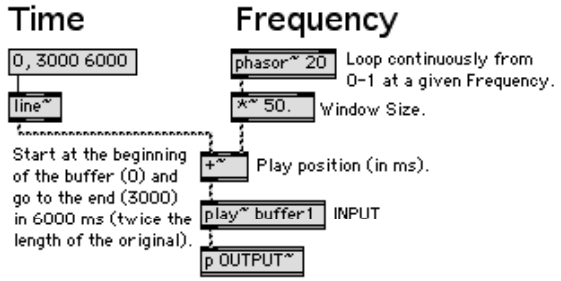

Figure 2a: Simplified grain engine with separate operators to control time and frequency.

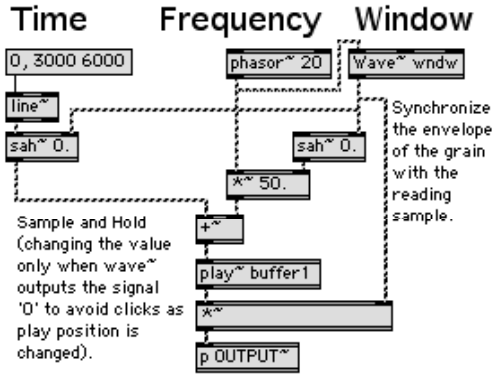

Figure 2b: Grain engine with windowing (from Nobuyasu Sakonda's Granular2.0).

Implementation Issues for Spontaneous-Input

Stationary vs. Circular Buffers:
One of the first major implementation decisions lies in whether to employ circular or stationary buffers-circular meaning a variable delay line or 'tapin~/tapout~' in MSP and stationary meaning an ordinary fixed buffer or 'buffer~' in MSP. Circular buffers, perhaps the most common for real-time audio applications, are easy to implement, use a negligible amount of CPU, and since they do not have fixed begin and end points, do not introduce discontinuities of the wave form. They do though present one major drawback in this particular application; since the older samples in the queueare constantly being replaced with newer samples, it becomes difficult, if not impossible to implement real-time time compression (as described below). Using a stationary buffer on the other hand allows for this possibility. Because the previously recorded samples remain in a stationary buffer they can also be repeated, (very handy for fine tuning parameters, both in the compositional process and during sound checks to for a particular environment) and can thus be subjected to further granularization. For these reasons, a stationary buffer was used with its recording cycle triggered by the performer on stage, typically with a MIDI foot pedal. The process of time expansion then begins 1 millisecond later (concurrently with the filling of the rest of the buffer). Note that the buffers are read with the MSP 'play~' object (see figure 2a and 2b) because of the necessity of that object being able to interpolate between samples and sample positions.

Implementation of Time Compression:
Obviously the process of real-time time compression, if implemented as above, would be over almost as soon as it started. Thus to enable the use this effect in a concert, the program imposes a delay before the compression begins. The program calculates this delay such that the time compression will end (the grains reaching the end of the buffer) 1 millisecond after the buffer is full. This has just the opposite effect from time expansion where the live and processed sounds begin virtually together and then *diverge*. In this case the sounds begin at very different points in time and then *converge*. Although other delay schemes are possible, this one seems to be the most effect in concert, as the end result sounds rather like a temporal crescendo.

Avoiding Clicks:
As Truax and others point out, the windowing of each grain element leaves little possibility for the production clicks or transients when reading a pre-recorded sound file, assuming that there are no discontinuities in the sound file itself. However, when one is "sampling on-the-fly" as Lippe puts it [Lippe 1994] there is a strong possibility that discontinuities will be recorded into the buffer, creating clicks that may then be read by multiple grain objects and subject to time stretching (not a very useful or pleasant sound for most). The use of a circular buffer is one obvious solution, but carries the drawbacks mentioned above. The best solution found for this application was to apply an amplitude envelope to the input source. Thus once the musician triggers a 'recording', the program initiates a 10-millisecond delay as the current output amplitude is ramped down to zero and the buffers are cleared. Then the input is ramped back to 1 again over another 10

milliseconds. As a slight time delay is natural between when the player physically triggers the recording and when they actually want the time compression/expansion to begin, the 10 to 20 millisecond time delay does not pose any problem in performance. As it is seldom aesthetically pleasing for the processed sound to end abruptly, a 100 millisecond downward ramping is applied to the end of the of the recording, which is musically satisfying for most circumstances.

Continuous Input/Output:
Another goal of the program was to have a system where you could continuously record and output; the musician would play and there would be a constant stream of time/pitch manipulated sound coming from the loudspeakers over a sustained period of time. The problem though comes as you reach the end of the buffer and loop back to the beginning. Even with a circular buffer (and thus no clicks) an audible temporal discontinuity will be obvious. To avoid this, the program employs two overlapping buffers with a time delay between them. The delay is calculated (multiplied times the expansion/contraction factor) such that one will hear only one 'image' of the sound (not to separate images) and thus with the ramping of the input source described above they create one smooth continuous output. Another benefit of this method is that when continuous recording is NOT required the other buffer can then be utilized for a second independent expansion/contraction rate, and thus one can also combine time expansion with time contraction simultaneously.

Graphic User Interface (GUI):
As mentioned above, it seemed crucial that a Graphical User Interface (GUI) give the musician all the information they needed and all the flexibility they wanted to use the technique effectively on stage. Chief among these are count down times for various elements. Figure 3 give the main control GUI. These include a countdown of "Remaining Recording Time". This is quite useful when you ask, for example, that a given chord be time stretched, and on top of this the player continues with the next passage. In this scenario, if the player starts the next passage too soon, it too will be time stretched, which many not be desired. Thus the "Remaining Recording Time" countdown tells the player exactly when the recording time ends (when the end of the buffer has been filled) and thus when they can continue. In such a scenario one may also wish to select the ringing of the chord but NOT the attack of the chord as material for time stretching. The ability to start the recording with a MIDI message from the player then allows the player to decide an appropriate offset time into the beginning of the sound event, often desired in studio applications. As figure 3 also shows, the main control window also counts down the remaining play times for both buffers (the buffer size times the expansion/compression factor plus compression delay if any).

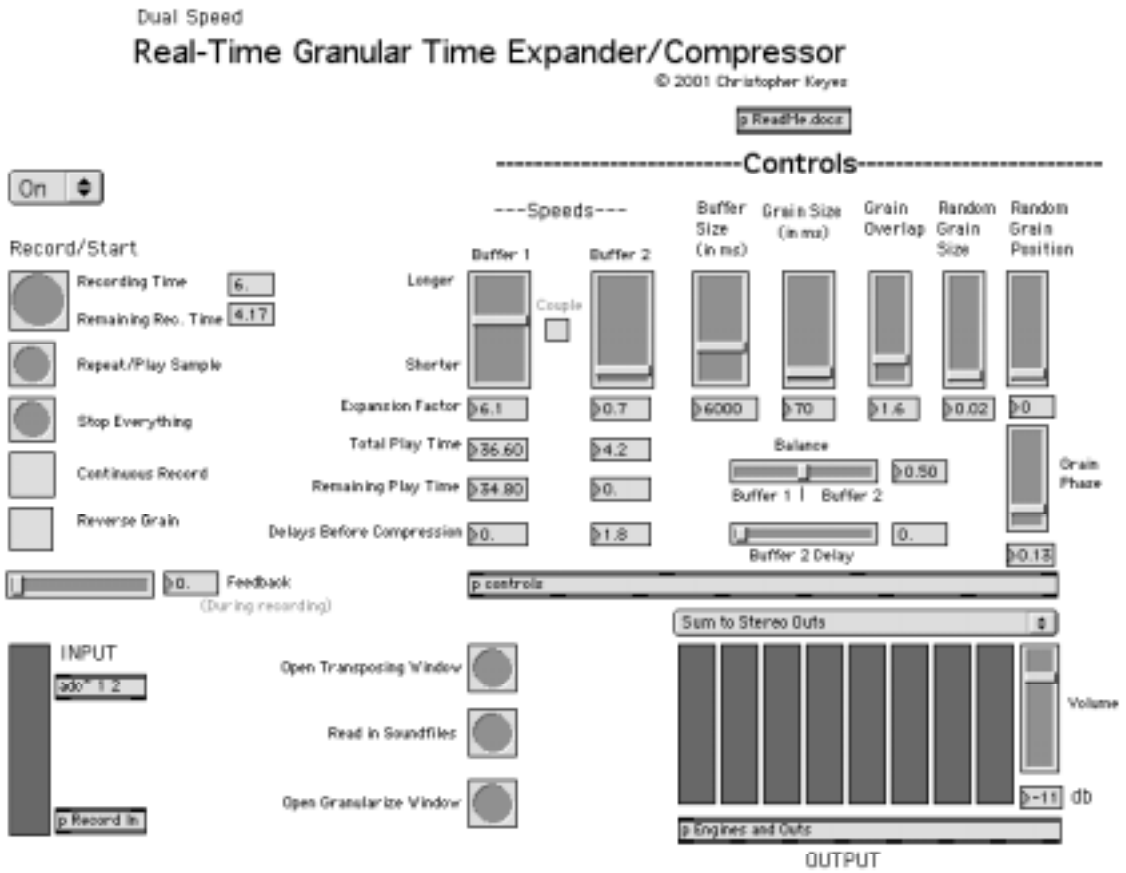

Figure 3: Main Control Window

Controlling Unwanted Modulation Effects:
In his article *Granular Synthesis of Sound* [Roads 1985] Curtis Roads warns of an amplitude modulation effect when grain sizes become small enough that the repetitions of the grains themselves becomes audible as a modulation signal. As can be seen in figure 3, the program provides a variety of grain parameter adjustments to avoid this. These include sliders to control the grain size, grain overlap, and to randomize the grain size. Randomizing the exact grain position is another very effective alternative, as is adjusting the relative phase of each of the grain engines.

Other Functions:
As the program utilizes 2 buffers each with 8 grains, the balance slider allows for effects where, for instance, one may desire a long time stretch in the background but a much louder time compression in the foreground. Reversing the direction of the phasors with the 'Reverse Grain' switch, and adding the output of the grains back into the input with the 'Feedback' slider achieve other effects. To expand the utility of the program the 'Read in Sound files' button does allow one to work with pre-recorded sounds, which is further enhanced by having two buffers. You can thus work with two different sounds simultaneously, or of course work with the same sound and two different expansion/compression rates. To work with more traditional granular sampling, another button opens the window shown in figure 4. With this window activated the position of

the grains previously supplied by the‘line~’ object is replaced by a start and end point in the buffer for continuous granularization of shorter sections.

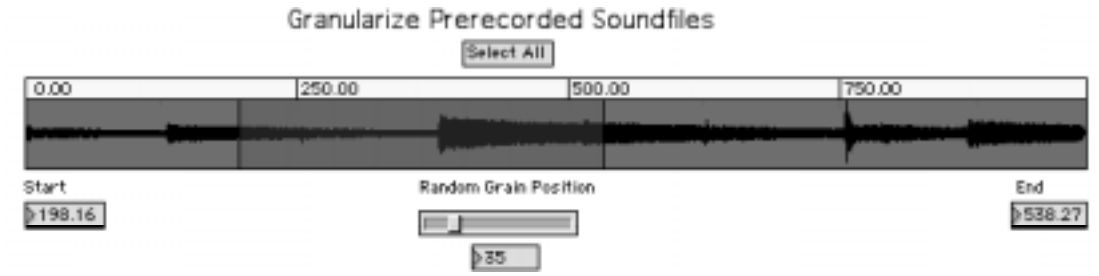

Figure 4: Granularization Window

Transposition:

Since frequency can be controlled independently from time, 16 grains can provide 16 different transpositions. To allow an intuitive control of transposition, the equation $2^{x*f/1200}$ was used such that when f=100 transposition is calculated in semitones (equal-tempered), 1/10 semitones when f=10, and by cents ($1/100^{th}$ of a semitone) when f=1. X then ranges from –64 to 64 yielding transpositions from +/-1 cent to +/-5 octaves or more. Figure 5 shows the exact MAX/MSP syntax.

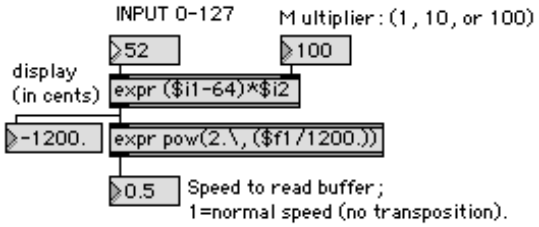

Figure 5: Transposition Equation in MAX/MSP Syntax

Figure 6 shows the GUI for the Transposition Window where buttons are used to change the 'f' factor above (1, 10, or 100) and sliders for each of the 16 grains. One can also “click and drag” the number boxes to the left of the sliders to control all sliders of each row. This function becomes especially useful when used with a MIDI controller. In my current composition I use a MIDI “expression pedal” to control all of the grain transpositions, half of them raising and the other half falling when I press the pedal. Thus with one motion I can control transposition of all 16 grains simultaneously. Lastly ‘Randomize Pitch’ and ‘Harmonize Pitch’ sliders make use of random number generators (also adapted from Nobuyasu Sakonda’s Granular2.0) to further effect the frequency of each of the grains.

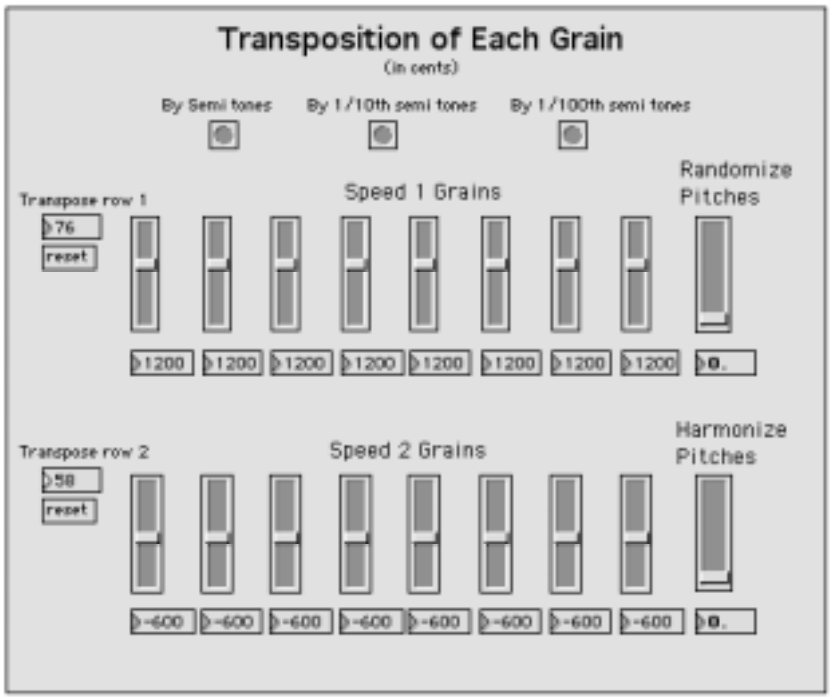

Figure 6: Transposition Window

Effects of Octophonic Spatialization:

The effects of multi-channel spatialization of each of the grains has turned out to be one of the most rewarding aspects of working with granular sampling. Allowing for grain sizes between 130-200 milliseconds and an expansion factor of 1 (no expansion or compression) leaves the sound relatively unaltered except for the amplitude envelope of each grain. When spatialized in stereo, the results do not seem especially interesting. When spatialized in 4 to 8 channels however, the effect becomes both interesting and yet subtle. Although there seems to be no real change in the sound, the amplitude envelop of each of the grains results in each channel having its own unique articulation of the sound, virtually simultaneous with the source (given only a 10-12 millisecond delay). Even longer grain sizes (1000-1500) and/or lessor distances between the grains produce effects similar to delay lines, but again, with separate amplitude envelops for each channel lending them a shimmering quality through the acoustic space. To maximize these effects and to minimize phase cancellation (resulting when grains become smaller, closer together, and of uniform size), the two grains for each channel are arranged such that the overlap factor is greatest between them. Thus with grain overlaps of 2 milliseconds, grain one is paired with grain 9 with its 18 milliseconds of distance for channel 1, grain 2 and 10 for channel 2 and so on.

End Product:

The finished product, a MAX4/MSP2 "patch" has so far accomplished the objectives set for it. Principally through time expansion, the program significantly alters the timbre of acoustical instruments input directly from a microphone, preserving their complex timbral characteristics while lending them an attractive time varying texture and spatialization with no audible delay. 13 years its junior, the system is (predictably) much

cheaper, has vastly more memory, and a more intuitive GUI than the one Barry Truax describes in his article of 1988 [Truax 1988]. It consumes roughly 55% of the available CPU cycles on a 500Mhz G3 processor, and roughly 32% of an 800Mhz G4 processor. Program can currently be downloaded from www.hkbu.edu.hk/~lamer/downloads

References:

Keyes, C. (1998) "The Ghost Within" *Christopher Keyes: Keyboard Works 1986-97* Centaur Records CRC2377, 8867 Highland Road, Suite 206 Baton Rouge, LA..

Keyes, C. (1999) "Li Jiang Etude No. 2 " *CD of the 1999International Computer Music Conference* (Beijing).

Lippe, C. (1994) "Real-Time Granular Sampling Using the IRCAM Signal Processing Workstation." *Contemporary Music Review* vol. 10:2, pp. 149-156.

Roads, C. (1978) "Automated Granular Synthesis of Sound" *Computer Music Journal* 2:2, pp. 61-62.

Roads, C. (1985) "Granular Synthesis of Sound" *Foundation of Computer Music* ed. C. Roads & J. Strawn 2:2, pp. 145-149.

Roads, C. (1988) "Introduction to Granular Synthesis" *Computer Music Journal* 12:2, pp. 11-13.

Roads, C. (1991) "Asynchronous Granular Synthesis" *Representations of Musical Signals* ed. by G. De Poli, A. Piccialli, and C. Roads Cambridge, Mass. : MIT Press.

Truax, B. (1988) "Real-TIme Granular Synthesis with a Digital Signal Processor" *Computer Music Journal* 12:2, pp. 14-26.

Truax, B. (1994) "Discovering Inner Complexity: Time Shifting and Transposition with a Real-time Granulation Technique " *Computer Music Journal* 18:2, pp. 38-48.