A NEW SCHEME FOR REAL-TIME LOOP MUSIC PRODUCTION BASED ON GRANULAR SIMILARITY AND PROBABILITY CONTROL

Pei Xiang

Center for Research in Computing and the Arts
University of California, San Diego
`pxiang@ucsd.edu`

ABSTRACT

In this paper, a new concept of real-time loop music production is introduced, along with its implementation in Pd. This scheme tends to improvise loop music based on very limited pattern (loop sample) materials. Four loops, each divided into 32 grains, work at the same time. Analysis of the spectral and energy similarities between every two grains are conducted, and the transition probability matrices generated by the analysis phase are consulted for each decision of grain choice during remixing. A joystick-style controller is designed to control the probability distribution, which changes the music characteristics in real-time.

While maintaining some characteristics of each loop pattern, the music generated by the program reveals a large space of variations and controllable improvisations. Real-time analysis is considered, that later will enable switching new patterns into the 4-pattern group during a performance. This scheme is a potential new method for live computer DJ mixing in the loop pattern level.

Sound examples, including four drum loops and the improvisations on them, can be found at

http://crca.ucsd.edu/˜pxiang/granuloop.htm

1. INTRODUCTION

As loop music comes to share the stage of pop music, various software has been developed, for sequencing (such as *Acid Pro*) or fine carving the loop samples (such as *ReCycle*). A loop based music, Drum 'n' Bass for example, may sound boring if it's only simply copies of a few repeating homogenous loop samples, with abrupt shifts into other loops at certain points. Often a preferred mixing is to have variations upon one loop, making every measure different from each other, while maintaining the cycling nature of the music. If this is done manually into every measure, it means a significant amount of work. Further, if there are requirements of real-time performances, it's impossible to realize with limited numbers of loop materials.

In the scheme introduced in this paper, each loop sample is broken down into 32 "grain" samples. After analysis, the energy and spectral similarities for every grain pair are calculated, and probability matrices are established for grains to find their possible "substitutions" when necessary. Details about the analysis and probability-controlling playback are described below.

2. THE ANALYSIS

The analysis involves three problems: 1. Separation: find the right positions in the sample to separate it into grains; 2. Energy weighting: find the normalized energy score that each grain has within its native loop. 3. Spectral similarity computation: determine the spectral distance between two grains from different loops.

2.1. Separation

In this implementation, drum loops are used. Grains are approximately equal length, but strictly dividing the drum loop into equal chunks will result in attack loss in the beginning of a grain or unwanted attack at the end, which should actually go to the next grain. In this work, this issue is left unaddressed. With the aid of *ReCycle* [1], peaks of drum events are successfully detected and separations are done manually. Below is a screen shot of the interface in *ReCycle* for separating loop c in the examples[1] into 32 grains.

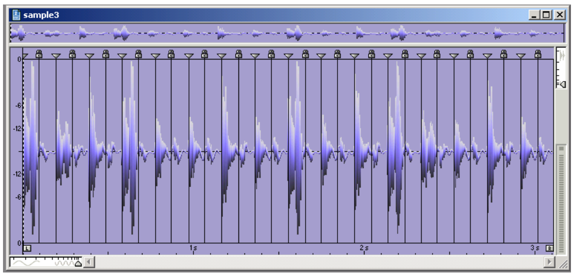

Figure 1: *ReCycle screen shot: separating loop c*

2.2. Energy Weighting

The average energy of each grain is calculated and normalized with the energy of the highest scored grain within each loop. Plots of the normalized energy scores for the four drum loops in the sound examples are presented in Figure 2. Strong and weak beats appear very clear in the plots.

2.3. Spectral Similarity Computation

Usually the way for examining drum samples [2] is to observe the attack, where most of the spectral characteristics are discoverable, and a lot of computation is saved. However, it doesn't work well on the grain "hits" here, as the resulting ratings of some similarities doesn't match the intuitive judgement from the ear at all. The reason might be that, although the grains still can be regarded as drum hits, their characteristic are more to be a wide band sound

---

[1] http://crca.ucsd.edu/˜pxiang/granuloop.htm

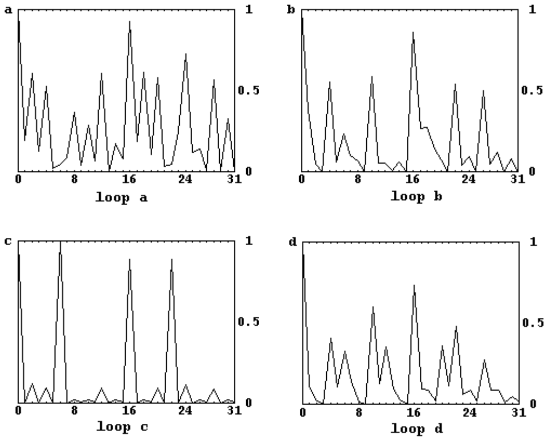

Figure 2: *Energy weighting plots*

event than a single percussion instrument hit, and this description works better on other kinds of loop patterns as well. So, a more thorough frequency analysis through a whole grain is needed.

In the implementation, for each grain, an 8192-point FFT is conducted, and the result is low-pass £ltered to get a smooth spectrum, which is sampled every 64 points to get vector $(x_1, x_2, \quad , x_n)$ (actually $n = 64$ here) as the *spectral vector* of the grain. The spectral similarity between two grains and are evaluated by the normalized inner product of their spectral vectors $(x_1, x_2, \quad , x_n)$ and $(\ _1, \ _2, \quad , \ _n)$ as follows:

$$= \frac{\qquad}{|\ |\ |\ |} = \frac{{}_i\, x_i \quad {}_i}{\overline{{}_i\, x_i^2} \quad \overline{{}_i \quad {}_i^2}} \tag{1}$$

where $i = 1, 2, \quad , n$. For identical vectors, the result is 1, with 0 being the smallest possible value.

Because is a normalized value, it is actually comparing the normalized spectral shapes. For a strong "downbeat" grain with bass drum and many other materials hit at the same time, energy is usually strong in a very broad band, and the spectrum is comparatively ¤at, with high energies. On the other hand, for a "£nishing" grain, the last grain of a beat or a loop for examples, it might not have a clear attack inside itself at all, and sometimes it is even partly buried in noise. The spectrum is also going to be ¤at, with low energy. These two grains are going to have a high score in according to equation (1), but, apparently, they are not suitable at all for each other as a "good" substitution. This is why energy weights are considered in addition to spectral similarity to calculate the £nal transition weights.

2.4. Transition Weights

For grains and , their *energy similarity* is de£ned as

$$= 1 \quad | \qquad | \tag{2}$$

where and are their normalized energy weights within the native drum loop, as discussed before. And a £nal *transition weight* T between them is de£ned as

$$T \quad = \tag{3}$$

In this way, grain pairs that have large energy differences OR spectral differences are prevented from getting a high score in their transition weights. With these transition weights, for every two drum loops, a 32 32 probability matrix is established. Four drum loops, labelled a, b, c and d, are chosen to form a group. Because of the structure of the controller, which will be discussed later, only four probability matrices are required here, for pairs (a, b), (b, c), (c, d) and (d, a).

3. PLAYBACK AND THE CONTROLLER

Figure 3 shows the controller for playback: Assume the length of the side of the square-shaped controller is d, imagine that the square is subdivided into four smaller squares sitting in the corners, with $d/2$-long sides. When the blue handle moves into any of the sub-squares, situations are similar.

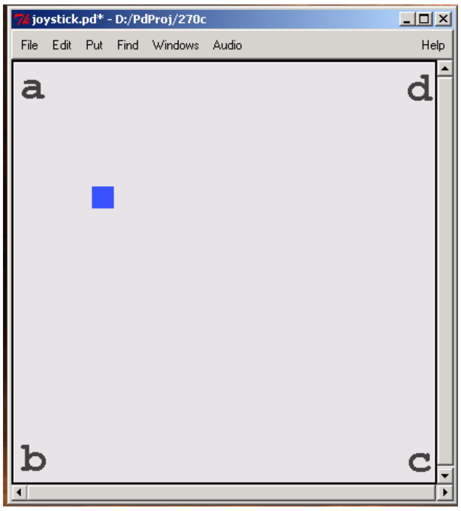

Figure 3: *Screen shot of the controller*

Take the northwestern square for example, calling it the "a" square. loop a is divided into grains $a_1, a_2, \quad , a_{32}$ in time order. When the handle is in this square, a cycling sequencer tries to loop-playback 32 grains in order. For the ith ($i = 1, 2, \quad$, 32) grain playback in each loop, the sequencer has three possible choices: 1. to playback a_i; 2. to substitute a_i with b_j ($j = 1, 2, \quad$, 32), a grain from b, based on the principle that the more similar b_j is to a_i, the more likely it is to be chosen; 3. same principle as for case 2, substitute a_i with d_j, a grain from the other neighboring square, loop d.

The probabilities of which of these 3 instances to choose is directly controlled by the handle. If x and are the horizontal and vertical distances from the handle to the upper-left corner, then P , P_b and P , which are the probabilities to choose instances 1, 2 and 3 described above, are calculated as:

$$P_b = \frac{\ }{d} \quad 0.5 \tag{4}$$

$$P \quad = \frac{x}{d} \quad 0.5 \tag{5}$$

$$P \quad = 1 \quad P_b \quad P \tag{6}$$

where d is the side length of the large square.

Similarly, groups $(P_{bb}, P_{b}\ \ , P_{b}\ \ )$, $(P\ \ \ , P_{\ b} P\ \ \ )$ and $(P\ \ \ ,$ $P\ \ \ , P\ \ \ )$ can be calculated for other three sub-squares, with some slight differences in the equations. These probabilities are calculated in real-time, to dynamically change the music character, which can be further described like: When the handle is near the "a" corner, music is mainly loop a, with very little variations using similar grains from b and d; as it moves away from "a" corner, more variations occur, and the proportion of grains used between b and d is depended on which neighboring sub-square the handle is closer to; when the handle goes to the center of the large square, but just within sub-square a, the music is totally the improvisation of grains from b and d based on the energy and spectral structure of loop a.

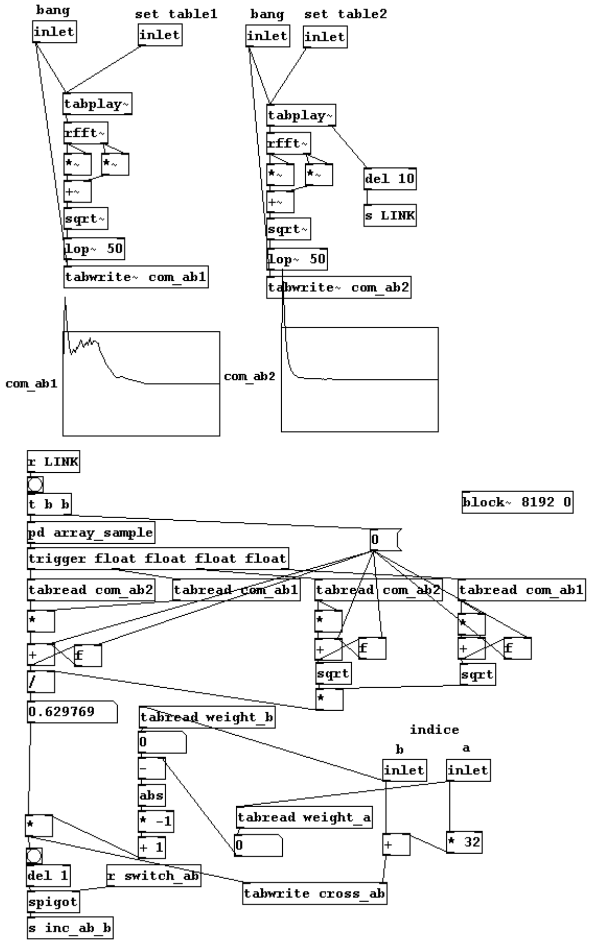

Figure 4: *Sub-patch for spectral similarity comparison*

4. IMPLEMENTATION NOTES

This scheme is implemented in Pd[2]. All the patches are available on the web site. Here's a simple explanation of the structure of the patch. Two top-level patches *master1_analysis.pd* and *master2_playback.pd* link and summarize all the rest, and separate the process into "analysis" and "playback" sections.

Analysis

The analysis is not done in real-time. The steps are: 1. Prepared grain samples are stored in tables independently; 2. Energy of each grain is measured by its average power - the rms of the grain, compared and normalized, then stored into tables like the ones in Figure 2; 3. Do the spectral comparison for each grain pair. One sub-patch for this process is shown in Figure 4.

As can be seen on the upper half of Figure 4, a pair of grain samples is put to a Fourier transformation (real part), then passed through a 50Hz low-pass filter. Two tables display their "smoothed" spectrums. The lower part of the Figure is the realization of Equations (1) through (3): *tabread* objects for *weight_a* and *weight_b*, according to proper indices, consult the and that generated in step 2, to get . Then, it is multiplied with the result of (number box 0.629769 in the figure) to get T . the object *tabwrite cross_ab* then writes the final result T into another table that holds the probability matrix between loops a and b. As stated before, there are 4 matrices of this kind in all, and the process on this patch should be repeated $32\quad 32\quad 4 = 4096$ times before the relationship between grains from the two loops become ready for playback. The $4096\quad 2$ times of 8192-point FFT computation is one of the reasons that this analysis is hard to implement in real-time in a patch work. After the four matrices are ready, the analysis is completed. Figure 5 shows the probability matrices produced in the patches. $32\quad 32$ matrices are presented in 1024-point-long arrays, to convenient the consultation when played back.

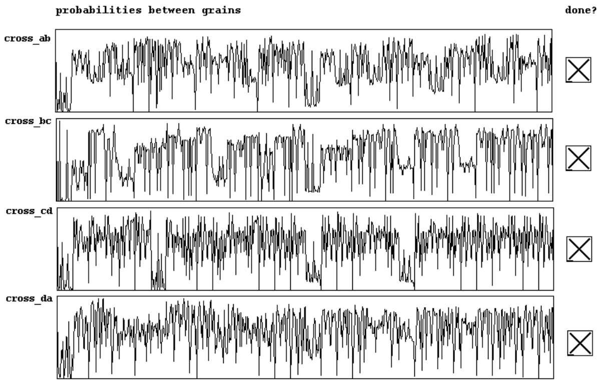

Figure 5: *Probability matrices presented in arrays*

Playback

Playback is a process mainly deals with random number generation and probability control by consulting the matrices as maps for probability relations.

Figure 6 shows one level of the patches that control the playback (for "a" area in Figure 3). In the patch, for a one-measure playback, each of the 32 instances triggers the following in order: First, send the current playback position within the measure to layers of some output patches; after receiving the location parameters generated by mouse-drags on the square controller (Fig-

---

[2] http://crca.ucsd.edu/~msp/software.html

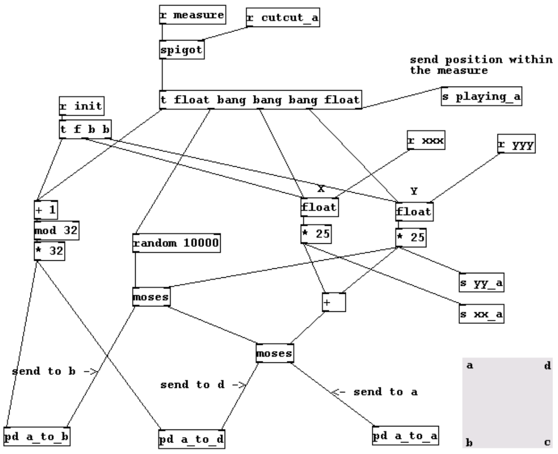

Figure 6: *One level of the patches for playback control*

ure 3), chances for choosing which group of grain samples (from loop a, b, or d on this patch) to play from are calculated (Equations (4) (6)), with the help of probability matrices related to this position and this sub-square area; then, a random number is generated to choose and trigger one grain to be played; finally, a new matrix is chosen, consulted and parameters are prepared for the next grain playback.

To make the grain substitutions more natural, a small amount of reverberation is added at the end of the playback route. It, to some extent, "glues" the grains into integrate cycles. Here, the simple *rev1˜* object, that is not a very good reverberator, is used, just for the purpose of simulation.

5. FUTURE WORKS

The Pd implementation just functions as one simulation for this idea. Much work can be done to improve the performance, add new ideas to this scheme, or make it an efficient production tool in the future.

5.1. More Reasonable Grain Separation

At the stage to separate loops into grains, the rigid rule here separates every loop into 32 approximately equal-length ones. If a drum hit has the length that occupies not just $1/32$ of the measure, but $1/16$ or more, the integrity of this event is harmed. The "2nd half", or, an incomplete ending section of this event is treated like another event with strange attacks, and to be played back somewhere later. To let each loop event keep its integrity, a more reasonable and logical way is not to separate it, but give it one kind of "length power" that valued, say, 2 out of 32. When it is chosen to be played back, a $2/32$ space within the measure should be ready for this event, for example. Of course, this requires a much more intelligent analyzer to deal with the raw loop samples and a more sophisticated playback control.

5.2. Make It Real-time

To make it really useful for live production, real-time requirements should be further satisfied. If it is made a software, implemented with lower levels of languages such as C++, the performance of the program should be greatly improved, and speed requirements for real-time analysis and control should not be a big problem.

What's more, in Figure 3, for each handle position, this design of controller makes possible use of grains from three "neighboring" loops on the square. This nature allows the fourth loop to be refreshed into a new one while none of its grains are supposed to be played back. During this time, it is possible to do analysis to the new coming sample and refresh related matrices. This makes the real-time DJ-style mixing possible, and never-ending new materials can be imported without stopping the music to do analysis.

5.3. The Reverberator

When computational potential is explored via other means of implementation, better reverberators will be capable to enhance the performance. Of course, to find a "good" reverberator has no specific rules. More, the parameters of the reverberator can also be linked to other control parameters, adjusting itself according to the playback tempo, nature of the grain, and some other factors, to make it a real dynamic "glue" for the grain re-combinations.

6. RESULTS AND CONCLUSIONS

Because the scheme is based on probability, the music generated has a controllable degree of randomness. Given a limited number of loop sample materials, the monotonous repetitions that result from lack of variety of raw materials are successfully avoided. Because the probability matrices are generated according to the spectral nature of grains, variations on loops turn out to be humanized and reasonable. As variation grains are chosen from neighboring loops, the interwoven cycles also make the transitions from one loop into another smooth and inventive.

In addition, structures such as opening, fill-in, cadence, and free improvisations in loop music could be generated with the controller, leaving space and freedom to the performer.

The most apparent advantage of this scheme is, you can use even only four 4-second-long loops to generate minutes of well structured music, with reasonable variations and coherency, and none of the measures identical to any other at the same time.

7. ACKNOWLEDGEMENT

Many thanks to Miller Puckette for introducing me the vector product idea in calculating spectral distance between two signals, and Ted Apel, for his help with paper submission and language refining.

8. REFERENCES

[1] Propellerhead Software http://www.propellerheads.se

[2] sound examples: http://crca.ucsd.edu/˜pxiang/granuloop.htm

[3] Puckette, M. and Apel, T., "Real-time audio analysis tools for Pd and MSP." *Proceedings, Internatioanl Computer Music Conference*, San Francisco: International Computer Music Association 1998, pp.109-112

[4] Moore, F.R., *Elements of Computer Music*. PTR Prentice Hall, 1990

[5] Blesser, B., "An Interdisciplinary Synthesis of Reverberation Viewpoints," *Journal of the AES*, Vol.49 No.10, 2001