EXPRESSIVE QUANTIZATION OF COMPLEX RHYTHMIC STRUCTURES FOR AUTOMATIC MUSIC TRANSCRIPTION

Mauricio Rodriguez
Superior Conservatory of Castile and Leon
marod@ccrma.stanford.edu

ABSTRACT

Two quantization models for 'expressive' rendering of complex rhythmic patterns are discussed. A multi-nesting quantizer captures expressivity by allowing fine-grained/high-quality resolution, thus covering the automatic transcription of a wide range of rhythmic configurations, yielding from simple to rather complex music notations. A look-up table quantizer is discussed as another model to attain expressivity and musical consistency; input is quantized by comparison of 'rhythmic similarity' from a user-defined data-set or look-up 'dictionary'.

Both quantizers are presented as computing assisting tools to facilitate the transcription of rhythmic structures into the symbolic domain (i.e. music notation).

Keywords: Computer Assisted Composition, Rhythmic Quantization, One-Level Quantizer, Multi-Level or Multi-Nesting Quantizer, Look-Up Table Quantizer, Accumulative or Sequential Quantizer.

1. INTRODUCTION

The process of quantizing numeric series representing onset/durations of rhythmic patterns, is a facility commonly employed in computer assisted composition environments to ease the rendering of input data into symbolic music representation (i.e. music notation). Robust algorithms for rhythmic quantization not only aim to produce an acceptable form of music notation, but to 'interpret' the input data in a categorical form while producing 'intelligent' and interesting musical results. An expressive quantizer model should take into consideration the former aspects, namely, it should allow for a logical representation to capture the way music material is structured while preserving a rather accurate and readable notational rendering of the input data, that ultimately, depicts the composer's notational intentions. Neither of these two features are easily achieved through the use of computing models when it comes to the problem of rhythmic quantization. There have been however, interesting quantizing models to categorize input data either taken from performance situations or generated through algorithmic processes. The Connectionist [1], the Bayesian [2] and the 'Kant-Quantizer' [3], are some of those effective quantizing models that logically "filter-out" and provide "structuring" of the input data to ease the rendering process of an automated music notation. Even though the logical data parsings of some of those aforementioned quantizers allow for effective music transcriptions, their notated results are generally circumscribed to rhythmic notations of a fairly conventional fashion. For instance, the connectionist quantizer is a model that operates through a *cell-network* system where an initial onset/duration set interacts with *activation cells* to gradually converge to an equilibrium state. The equilibrium state seeks for simple-ratios between adjacent durations of the initial values from the onset/duration sequence; if no simple tuplet-ratios are found, the activation mechanism adjusts the original sequence until its values are rounded to evenly complete a subdivided beat. Within this mechanism, an irregular tuplet (if found) must follow an equivalent rhythmic figure until the full subdivided beat is completed, voiding the case of sequentially having, for instance, one irregular tuplet after another irregular tuplet of a different species, which is the core principle of a multi-nested rhythmic notation.

Most professional "general-purpose" quantizers, such as the "omquantify" (Open Music), the "gquantify" (PatchWork & PWGL) and the "ksquant" (PWGL) among others, treat their input data as linear arrays of numbers where durations follow to the next ones without determining any relational scheme between them. This principle, instead of being a disadvantage, permits to generalize and apply the quantization process to the most varied sets of data, from arbitrarily defined inputs to different algorithmic-generated numeric values. As a previous step for final notation output, the user of these quantizers can calibrate some quantization parameters, thus facilitating a more personal notational result; however, the limitations of "general-purpose"

quantizers lacking the supervising of "logical/intelligent" algorithms or user input inspection, are evident when, for instance, trying to quantize a simple rhythmic pattern of a *ritardando* figure, whose notational result would be most likely quantized with lots of tied irregular tuplets, without this resulting quantization necessarily showing the simple gesture-figure of a deceleration.

A general-purpose multi-nesting quantizer would not necessarily overcome the limitations shared by previous "non-logical" quantizers, however, it is claimed that a refined level of musical expressivity is achieved when the problem of quantization is generalized to capture and render complex rhythmic structures such as those present in multi-nested (fine-grained) rhythmic patterns.

2. XA-LAN MULTI-LEVEL QUANTIZER

Xa-Lan is a LISP-based software to generate expressive music scores based on graphic transformations that control the different symbolic/notational elements of a music score [4]. Xa-Lan relies on three modules (or engines) to produce output. The third of these modules is a multi-level or multi-nesting quantizer that allows to transcribe the onsets and durations of rhythmic patterns into accurate, complex, and expressive rhythmic notations.

The multi-nesting quantizer is a recursive adaptation of a simple one-level quantizer [5]. The algorithm of the one-level quantizer compares equal subdivisions of a given temporal segment to the original onset-duration sequence, searching for minimal-error differences. Since larger number of subdivisions reduce error-difference values, the minimal-error curve is compared to an ideal fitting curve whose maximum difference is chosen as the earliest optimal quantization (Figure 1).

The multi-level quantizer in Xa-Lan recursively applies the former quantizing process to different portions of a temporal segment, from the measure level to the beat and the beat subdivision levels, therefore, yielding nested rhythmic figures when necessary. The Xa-Lan multi-nesting quantizer is unique in its functionality. No other quantizer available in common computer-assisted composition environments, such as PatchWork, OpenMusic, Symbolic Composer, or PWGL (to name some), can render automated nested rhythmic figures to allow a rather refined quantizing resolution. The Xa-Lan quantizer can also be aligned to any user-defined metrics-grid that works as a "structural container" of the quantization. The maximum subdivision for irregular tuplets at any nesting level, can also be arbitrarily set by the user, allowing for different notational resolutions of the same input.

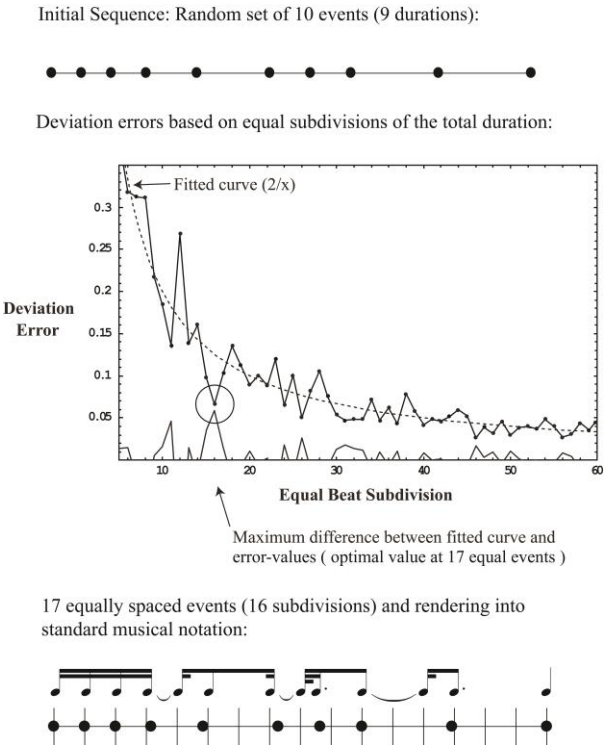

Figure 1. One-Level Quantizer.

To dynamically interact with the facilities of the multi-level quantizer, Xa-Lan uses the "Expressive Notation Package" (ENP) from the visual language of PWGL [6] for final display (Figure 2).

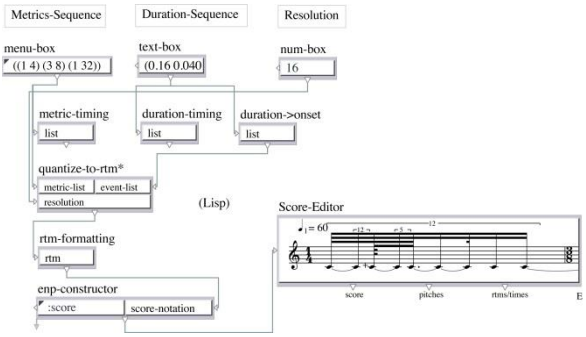

Figure 2. Multi-Level Quantizer interface on ENP.

To observe some of the possible results that can be obtained by this mutli-nesting quantizer, consider the following duration sequence: 0.16 0.3 0.25 0.040000007 0.58000005 0.37 0.45000005 0.29999995 0.17499995. This array of numbers will be quantized using the following arbitrary metric container: 1/4, 3/8, and 1/32. The maximum number of subdivisions per nesting level is also arbitrarily set to 12 (Figure 3):

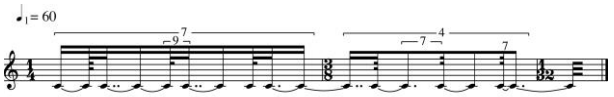

Figure 3. Multi-Level Quantization I.

In the following figure, the resolution subdivision is downsampled by half (to 6) of the previous quantization (Figure 4) :

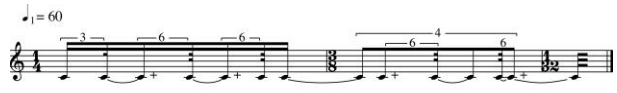

Figure 4. Multi-Level Quantization II.

Lastly, the same array of durations are quantized with a different metric container (6/32, 1/4, and 7/32) and the maximum number of subdivisions per nesting level is 12 again (Figure 5) :

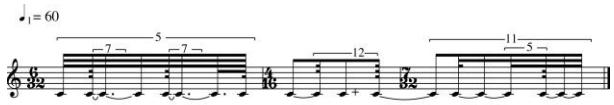

Figure 5. Multi-Level Quantization III.

3. LOOK-UP TABLE QUANTIZER

Another quantization model that attempts to capture complex rhythmic structures and render them into a meaningful and expressive musical context, is through the use of a look-up table quantizer. In a look-up quantizer, a 'dictionary' with predefined rhythmic-patterns uses each of its entries as place-holders where input onset/durations sequences are sieved-on, choosing the place-holders or "rhythmic grids" with minimal error-difference as the best quantized approximations. The real advantage of a look-up table quantizer is that the user can define a unique and precise dictionary of rhythmic configurations from where quantization takes place, ensuring an idiomatic behavior that easily conforms to a compositional system of temporal organization.

The internal workings of the look-up quantizer are similar to the ones of the multi-level quantizer, except that the searching space to compare and get the optimal error- difference is manually introduced by the user, instead of being algorithmically generated. Once the user includes a new "rhythmic word" in the dictionary, by using a symbolic "rhythmic-tree" representation, the first task for the look-up algorithm is to convert any "word" into its equivalent timing equivalent (e.g. (1 (1 1 (2 (1 1 1))) is equivalent to 0.25, 0.25, 0.5/3, 0.5/3 and 0.5/3 seconds, assuming a quarter-note is equal to one second). From there, the comparison of the original time-input sequence with the time-converted "words" is straight. The next step is to do the proper rhythmic configuration groupings of the "word" that is chosen as optimal quantization. If for instance, the original time input is 0.25, 0.58 and 0.17 seconds, the place-holder word of (1 (1 1 (2 (1 1 1))) would be output as (1 (1 1 (2 (2.0 1))), being this result the best quantization among the given words of that dictionary.

The idea of a user-predefined rhythmic dictionary might appear burdensome at first, but this quantizing model is essentially as effective as any other general purpose quantizer, with the invaluable advantage of rendering rhythmic results that fully conform to a precise selection of rhythmic configurations that are previously input by its users, and therefore, the expressivity of the resulting transcriptions completely accommodate to the idiomatic and aesthetic needs of composers.

```
(defvar *rhythm-reservoir*
'(
(1)
(1 1)
(1 1 1)
(1 1 1 1)
(1 1 1 1 1)
(1 1 1 1 1 1)
(1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1)
(1 (7 (1 1)))
(1 (7 (1 1 1)))
(1 (7 (1 1 1 1)))
(1 (7 (1 1 1 1 1)))
(1 (7 (1 1 1 1 1 1)))
((7 (1 1)) 1)
((7 (1 1 1)) 1)
((7 (1 1 1 1)) 1)
((7 (1 1 1 1 1)) 1)
((7 (1 1 1 1 1 1)) 1)
(1 (6 (1 1 1 1)) 1)
(1 1 (6 (1 1 1 1)))
((6 (1 1 1 1)) 1 1)
(1 (6 (1 1 1 1 1)) 1)
(1 1 (6 (1 1 1 1 1)))
((6 (1 1 1 1 1)) 1 1)
(1 1 1 (5 (1 1)))
(1 1 1 (5 (1 1 1)))
(1 1 1 (5 (1 1 1 1)))
((5 (1 1)) 1 1 1)
((5 (1 1 1)) 1 1 1)
((5 (1 1 1 1)) 1 1 1)
(1 (5 (1 1)) 1 1)
(1 (5 (1 1 1)) 1 1)
(1 (5 (1 1 1 1)) 1 1)
(1 1 (5 (1 1)) 1)
(1 1 (5 (1 1 1)) 1)
(1 1 (5 (1 1 1 1)) 1)
((3 (1 1)) (5 (1 1)))
((3 (1 1)) (5 (1 1 1)))
((3 (1 1)) (5 (1 1 1 1)))
((5 (1 1)) (3 (1 1)))
((5 (1 1 1)) (3 (1 1)))
((5 (1 1 1 1)) (3 (1 1)))
(1 1 (4 (1 1 1)) 1 1)
(1 1 (3 (1 1)) (3 (1 1)))
((3 (1 1)) (3 (1 1)) 1 1)
((3 (1 1)) 1 1 (3 (1 1)))
(1 1 1 (3 (1 1)) 1 1)
(1 1 (3 (1 1)) 1 1 1)
(1 (3 (1 1)) 1 1 1 1)
(1 1 1 1 (3 (1 1)) 1)
(1 (3 (1 1)) (4 (1 1 1)))
((4 (1 1 1)) (3 (1 1)) 1)
(1 (4 (1 1 1)) 1 1 1)
(1 1 1 (4 (1 1 1)) 1)
(1 (4 (1 1 1)) (3 (1 1)))
((3 (1 1)) (4 (1 1 1)) 1)
((3 (1 1)) 1 (4 (1 1 1)))
((4 (1 1 1)) 1 (3 (1 1)))
((4 (1 1 1)) 1 1 1 1)
(1 1 1 1 (4 (1 1 1)))
))
```

Table 1. User-defined 'rhythmic-grid' dictionary

When working with the look-up table quantizer, it is important to keep in mind that varied and fine-grained quantizations can only take place if there is a comprehensively large data-set of place-holder rhythmic words in the dictionary, otherwise one or several input values in some cases could not be quantized, in which case, the output of the algorithm will indicate the number of non-quantized events. An interesting compositional strategy to use this quantizer

can be forcing the quantization process to a limited set of dictionary-words, and by gradually changing, or rather expanding the searching space where quantization takes place, different resolutions of the quantization would show the kind of transcriptions that fit more naturally to the original input data. To facilitate this compositional methodology, there is an additional routine in this quantizer to compare and sort the deviation-error similarity (in an ascending to descending order) of one chosen word in relation to all the other words of the dictionary. Additionally, the similarity comparison among words can be truncated to show only the ones that present the same "rhythmic profile" from a reference word being compared; for example, the rhythmic tree (1 (3 1 2)) could be output as similar to (1 (4 1 3)) since both share the same "rhythmic profile", meaning that the first duration of the group is larger than the second, and the second being shorter than the third. The following figures show the similarity rankings from the rhythmic tree (1 (2 1 1 4)), which is indeed a grouped version of the simple rhythmic tree (1 (1 1 1 1 1 1 1 1)); first, similarity is presented regardless rhythmic profile (Figure 6), and then truncated to show words with equivalent profiles (Figure 7). The results of these comparisons are based on the following searching-space dictionary:

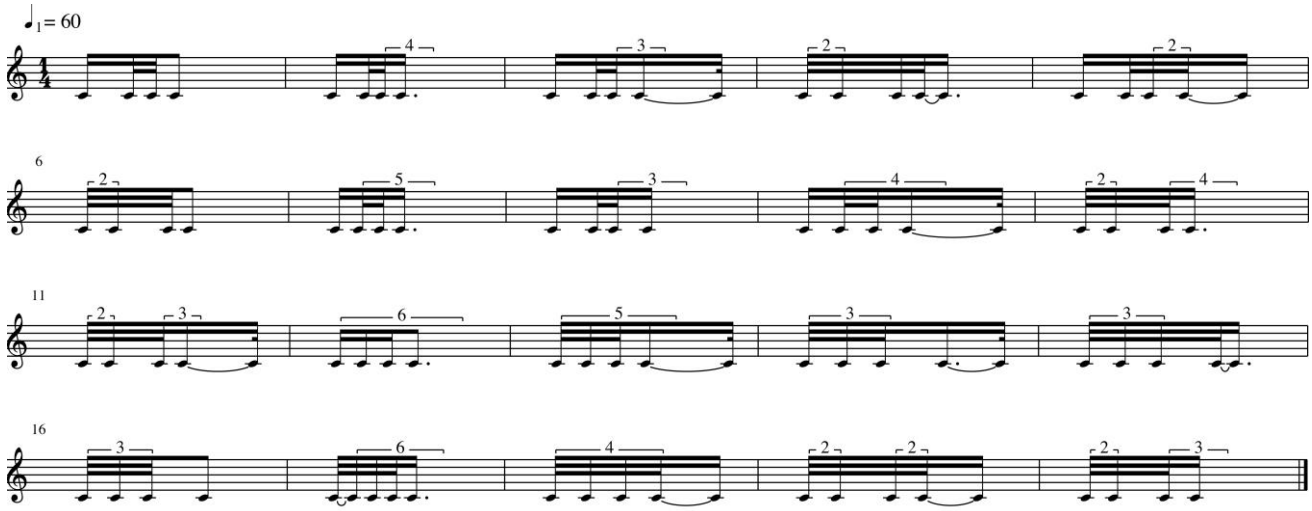

Figure 6. Rhythmic similarity

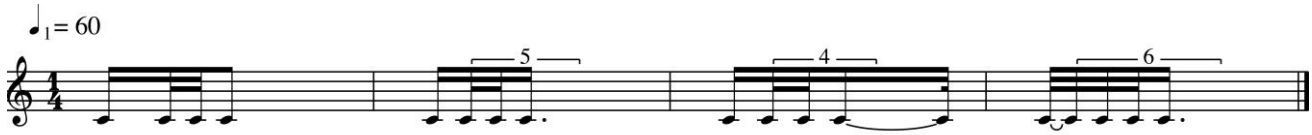

Figure 7. Rhythmic similarity with equal ‘profile’.

Further implementations

In order to make more flexible and expressive the quantized results of the look-up table quantizer, a changing metrics-grid that would serve as a variable structural container, could enhance the quantization results as it happens with the metric flexibility already implemented on the multi-nesting quantizer. Another interesting feature to implement could be the automatic or algorithmic generation of additional dictionary-words or “rhythmic grids”, based on the analysis of rhythmic similarities equivalent to those manually defined. Lastly, an additional feature to consider in any real expressive quantizer, is the possibility to render exceptional notation cases, as it happens when “incomplete” subdivisions of the beat are sequentially linked or concatenated, without necessarily rounding with the reminders of the previous tuplet subdivisions. The results of such an "accumulative" or "sequential quantizer" would reduce the compromising filtering of the input data that occurs during the output of a general-purpose quantizer.

4. CONCLUSIONS

Automatizing the rendering of an *expressive* rhythmic notation is a task that demands, on the one hand, a logical structuring or shaping of the input data to

provide the resulting notation with musically consistent results, and on the other hand, quantization results should conform to the aesthetic and notational idiosyncrasy of a given user. Two general-purpose quantizing models have been presented to aim for notational expressivity from different perspectives. A multi-level or multi-nesting quantizer achieves 'expression' by fine-grained / high-quality resolution output, while preserving an uncompromising general-purpose applicability (i.e. no pre/post filter processes are applied to input data). A look-up table quantizer guarantees expressivity through a user-defined data-set that works as a closed searching-space from where quantization takes place. These two quantizers aim to be used as computing tools to facilitate and assist the composition *and writing* or notational rendering of music works.

5. REFERENCES

[1] P. Desain, and H. Honing, "Quantization of musical time: a connectionist approach" in *Music and Connectionism*, MIT Press Cambridge, 1991, pp. 150-167

[2] A. T. Cemgil, P. Desain, and B. Kappen, "Rhythmic Quantization for Transcription", Computer Music Journal, vol. 2, n^{o}. 24, 2000, pp. 60-76.

[3] C. Agon, G. Assayag, J. Fineberg, and C. Rueda, "Kant: A critique of pure quantification", in *Proceedings of the International Computer Music Conference*, *International Computer Music Association*, Aarhus Denmark, 1994, pp. 52–9.

[4] M. Rodriguez, "Xa-lan: Algorithmic Generation of Expressive Music Scores Based on Signal Analysis and Graphical Transformations" in *Proceedings of the International Workshop on Musical Metacreation - 8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Stanford University, 2012, pp. 83-85.

[5] C. Saap, "Rhythmic Quantizer", *unpublished paper*, Center for Computer Assisted Research in the Humanities (CCARH), Stanford University, 2011.

[6] M. Laurson, and M. Kuuskankare, "PWGL: A Novel Visual Language based on Common Lisp, CLOS, and OpenGL, in *Proceedings of the International Computer Music Conference*, San Francisco, 2002, pp. 142-145.