ANALYSIS BY COMPRESSION: AUTOMATIC GENERATION OF COMPACT GEOMETRIC ENCODINGS OF MUSICAL OBJECTS

David Meredith
Aalborg University, Denmark
dave@create.aau.dk

ABSTRACT

MEL is a geometric music encoding language designed to allow for musical objects to be encoded parsimoniously as sets of points in pitch-time space, generated by performing geometric transformations on component patterns. MEL has been implemented in Java and coupled with the SIATEC pattern discovery algorithm to allow for compact encodings to be generated automatically from *in extenso* note lists. The MEL-SIATEC system is founded on the belief that music analysis and music perception can be modelled as the compression of *in extenso* descriptions of musical objects.

1. INTRODUCTION

Let a *musical object* be any quantity of music, whether it be a motiv, a phrase, a voice, a chord, a movement, a work, a corpus or even some very large quantity of music such as "all Western tonal music" or even "all music". The activities of music analysis and listening to music share the common goal of finding the best possible explanations for musical objects. The work presented here is founded upon the belief that this goal is equivalent to that of compressing musical objects as much as possible. Indeed, the view adopted here is that the field of music theory can be characterised as the search for concepts that allow for as much music as possible to be described as parsimoniously as possible.

On this view, music analysis and musical listening can each be modelled as a process that takes an *in extenso* representation or description of a musical object as input and outputs a compact encoding that can be used to generate or reconstruct the input object: the more compact the encoding, the better the musical object has been understood or explained. Both the input and the output of such a process are descriptions, but if the output is shorter than the input, then it constitutes at least a partial explanation of the input, since, in order to achieve any compression of the input, some structure needs to have been recognized in it. Algorithmic information theory [1] tells us that, if there is no structure in the input (i.e., the input is algorithmically random), then no compression is possible. Conversely, the more structure there is in the input, the more compactly it can be described.

The brains of music analysts and listeners are not perfect compressors, so they are typically unable to find the *most* compact encodings possible for any given input. In fact, algorithmic information theory tells us that, in general, it is impossible to confirm that a given encoding of an object is the shortest possible, since the *algorithmic* (or *Kolmogorov*) *complexity* of an object is not computable [1]. Moreover, the actual encoding generated by an individual's brain for a given musical object will depend upon what previous objects that individual's brain has encoded (i.e., compressed) and even the *order* in which those objects were processed. This is because recognizing a relationship between a part, *X*, of an input and a part, *Y,* of an encoding of some previously processed input allows *X* to be described simply by encoding the relationship that maps *Y* onto *X*, which can often lead to a compressed representation of *X*. It seems plausible that this greedy strategy that prefers to represent a new input in terms of elements of encodings of previous inputs, is adopted by the brain when it analyses or listens to music. It is easy to see how such a mechanism would result in the differences that arise between the ways that individual listeners and analysts understand the same musical objects.

2. MEL: A GEOMETRIC MUSIC ENCODING LANGUAGE

The work presented here relates closely to psychological *coding theories* of perceptual organization that employ the *minimum* or *simplicity principle* [2]. In the coding theory approach, a *coding language* is devised to represent the possible structures of patterns in a particular domain. The preferred organizations (i.e., the ones that the theory predicts will be perceived) are then the ones that have the shortest encodings in the language. Coding theories of this type have been proposed to explain the perception of serial patterns, visual patterns and musical patterns [3]. MEL, the language proposed here, generalises and extends the music coding language of Deutsch and Feroe [3] by adopting a geometric approach, along the lines of that proposed by Meredith *et al*. [4]. In MEL, a passage of polyphonic music is represented as a set of multidimensional points, generated by performing geometric transformations on component patterns. The language introduces the concept of a *periodic mask*, a gener-

alisation of Deutsch and Feroe's notion of a pitch alphabet. Such masks can be applied to the time dimension to represent parsimoniously the hierarchical structures of rhythms and metres. In the pitch dimension, masks can be applied in an identical fashion to represent the hierarchical structures of "pitch alphabets" such as scales and chords.

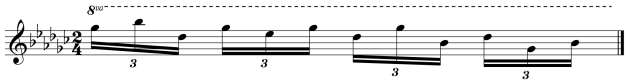

Figure 1 The right-hand of the first bar of Chopin's *Étude* in G flat major, Op. 10, No. 1.

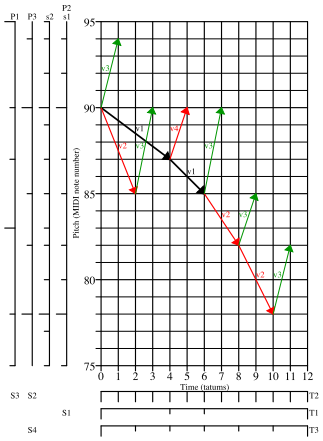

Figure 2 A graphical representation of the structure encoded in Figure 3. The vertical lines with tick marks on the left and the horizontal lines with tick marks under the graph represent masks in pitch and time.

```
MEL25;
n1 = note(0,90);                    //First note
p = coords(1,-1);                   //Corresponds to p ("previous") operator in Deutsch and Feroe
n = coords(1,1);                    //Corresponds to n ("next") operator in Deutsch and Feroe
ms1 = maskStructure(2,2,3);         //Triad mask structure
s1 = mask(6,2,2,3,2,3);             //Background scale: Gb pentatonic
s2 = mask(6,2,2,1,2,2,2,1);         //Gb major scale
T1 = maskSequence(mask(0,4,2,6));   //Background rhythm
T2 = maskSequence(mask(0,1));       //Tatum time mask sequence
T3 = maskSequence(mask(0,2));       //Time mask sequence for alternate semiquavers
P1 = maskSequence(s2,mask(3,ms1));  //Subdominant triad in Gb major
P2 = maskSequence(s1);              //Pitch mask sequence for background (Gb pentatonic)
P3 = maskSequence(s2,mask(0,ms1));  //Tonic triad in Gb major
S1 = space(T1,P2);                  //Background space
S2 = space(T2,P3);                  //Space for first four semiquavers
S3 = space(T2,P1);                  //Space for vector v4
S4 = space(T3,P3);                  //Space for vector v2
v1 = vector(p,S1);                  // \
v2 = vector(p,S4);                  //  | Vectors - see figure ->
v3 = vector(n,S2);                  //  |
v4 = vector(n,S3);                  // /
Q1 = repeat(2,v1);                  //Sequence of 2 v1 vectors in background space
Q2 = repeat(2,v2);                  //Sequence of 2 v2 vectors in middleground space
R1 = product(v2,v3);                //Cartesian product of v2 and v3
R2 = product(Q2,v3);                //Cartesian product of Q2 = <v2,v2> and v3
add(translate(n1,
            product(Q1,                     //<v1,v1>
                    sequence(R1,            //v2 x v3
                             vectorSumSet(v4),  //v4
                             R2))));        //<v2,v2> x v3
```

Figure 3 An encoding in MEL of the musical object in Figure 1.

Figure 3 shows an encoding in MEL of the musical fragment in Figure 1. Figure 2 represents graphically how the set of notes in Figure 1 is generated by multiple translations of component patterns within "masked spaces". The encoding in Figure 3 is not particularly short because it includes definitions of rhythms, metres and pitch alphabets that could be re-used to produce a parsimonious encoding of a much larger set of tonal musical objects.

3. AUTOMATIC GENERATION OF SHORT ENCODINGS USING SIATEC

The MEL language has been implemented in Java and incorporates an efficient implementation of an adaptation of Meredith *et al.*'s SIATEC pattern discovery algorithm [4]. This algorithm takes as input an *in extenso* MEL encoding of a musical object in which all the notes are explicitly stated and compresses it to produce a compact encoding in terms of translational equivalence classes (TECs) of maximal translatable patterns (MTPs) (see [4]). The TECs selected are those that produce the best compression of the input and that contain patterns that have high "compactness" (i.e., their bounding boxes do not contain many non-pattern notes). As an example, Figure 4 shows the best TEC found by the MEL implementation of SIATEC for a fugue by J. S. Bach.

4. REFERENCES

[1] M. Li and P. Vitányi: *An Introduction to Kolmogorov Complexity and Its Applications.* (3rd Edition), Springer: Berlin, 2008.

[2] J. R. Pomerantz and M. Kubovy: "Theoretical approaches to perceptual organization: Simplicity and likelihood principles," in: K. R. Boff, L. Kaufman, & J. P. Thomas (Eds.), *Handbook of perception and human performance: Volume II*, Chapter 36. Wiley: New York, 1986.

[3] D. Deutsch and J. Feroe: "The internal representation of pitch sequences in tonal music," *Psych. Rev.*, Vol. 88, No. 6, pp. 503–522, 1981.

[4] D. Meredith, K. Lemström, and G. A. Wiggins: "Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music," *J. of New Music Res.*, Vol. 31, No. 4, pp. 321–345, 2002.

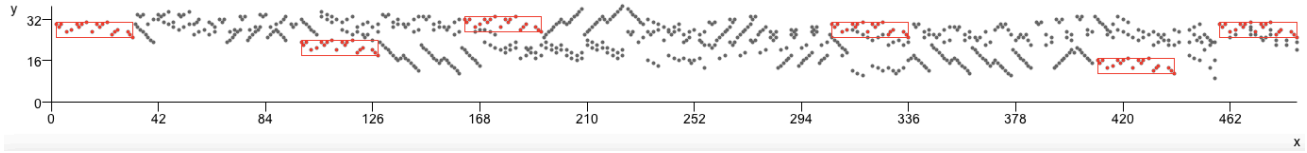

Figure 4 A graphical representation of the best TEC (indicated in red boxes) found by the version of SIATEC implemented in MEL for the Fugue in C minor from Book 1 (BWV 847) of J. S. Bach's *Das Wohltemperirte Clavier*. This TEC is the one that achieves the best compression ratio over the notes covered by it. It also has a maximally compact pattern (i.e., the bounding box of the first occurrence of the pattern does not contain any non-pattern points).