SYNPY: A PYTHON TOOLKIT FOR SYNCOPATION MODELLING

Chunyang Song
Queen Mary, University of London
dr.chunyang.song@gmail.com

Marcus Pearce
Queen Mary, University of London
marcus.pearce@qmul.ac.uk

Christopher Harte
University of York
christopher.harte@york.ac.uk

ABSTRACT

In this paper we present SynPy, an open-source software toolkit for quantifying syncopation. It is flexible yet easy to use, providing the first comprehensive set of implementations for seven widely known syncopation models using a simple plugin architecture for extensibility. SynPy is able to process multiple bars of music containing arbitrary rhythm patterns and can accept time-signature and tempo changes within a piece. The toolkit can take input from various sources including text annotations and standard MIDI files. Results can also be output to XML and JSON file formats.

This toolkit will be valuable to the computational music analysis community, meeting the needs of a broad range of studies where a quantitative measure of syncopation is required. It facilitates a new degree of comparison for existing syncopation models and also provides a convenient platform for the development and testing of new models.

1. INTRODUCTION

Syncopation is a fundamental feature of rhythm in music and a crucial aspect of musical character in many styles and cultures. Having comprehensive models to capture syncopation perception allows us to better understand the broader aspects of music perception. Over the last thirty years, several modelling approaches for syncopation have been developed and widely used in studies in multiple disciplines [1–8]. To date, formal investigations on the links between syncopation and music perception subjects such as meter induction [9, 10], emotion [8], groove [11, 12] and neurophysiological responses [13, 14], have largely relied on quantitative measures of syncopation. However, until now there has not been a comprehensive reference implementation of the different algorithms available to facilitate quantifying syncopation.

In [15], Song provides a consolidated mathematical framework and in-depth review of seven widely used syncopation models: Longuet-Higgins and Lee [1], Pressing [2, 16], Toussaint's Metric Complexity [3], Sioros and Guedes [4, 17], Keith [5], Toussaint's off-beatness measure [6] and Gómez et al.'s Weighted Note-to-Beat Distance [7]. With the exception of Sioros and Guedes' model, code for which was open-sourced as part of the Kinetic project [18], reference code for the models has not previously been publically available. Based on this mathematical framework, the SynPy toolkit (available from the repository at [19]) provides implementations of these syncopation models in the Python programming language.

The toolkit not only provides the first open-source implementation of these models in a unified framework but also allows convenient data input from standard MIDI files and text-based rhythm annotations. Multiple bars of music can be processed, reporting syncopation values bar by bar as well as descriptive statistics across a whole piece. Strengths of the toolkit also include easy output to XML and JSON files plus the ability to accept arbitrary rhythm patterns as well as time-signature and tempo changes. In addition, the toolkit defines a common interface for syncopation models, providing a simple plugin architecture for future extensibility.

In Section 2 we introduce mathematical representations of a few key rhythmic concepts that form the basis of the toolkit then briefly review seven syncopation models that have been implemented. In Section 3 we outline the architecture of SynPy, describing input sources, options and usage.

2. BACKGROUND

In this section, to introduce the theory behind the toolkit, we briefly present key aspects of its underlying mathematical framework (described in detail in [15]) and then give a short overview of each of the implemented syncopation models.

2.1 Time-span

The term *time-span* has been defined as the period between two points in time, including all time points in between [20]. To represent a given rhythm, we must specify the time-span within which it occurs by defining a reference time origin t_{org} and end time t_{end}, the total duration t_{span} of which is $t_{\text{span}} = t_{\text{end}} - t_{\text{org}}$ (Figure 1).

For the SynPy toolkit, we use *ticks* as as the basic time unit as opposed to seconds (in keeping with the representation used for standard MIDI files) where the rate is given in *Ticks Per Quarter-note* (TPQ). The TPQ rate that is chosen is arbitrary so long as the start time and duration of all notes in a rhythm pattern can be represented as integer tick values. As Figure 2 demonstrates, the *Son* clave rhythm pattern could be correctly represented both at 8 and 4 TPQ but not at 2 TPQ because the pattern contains a note that starts on the fourth 16th-note position of the bar.

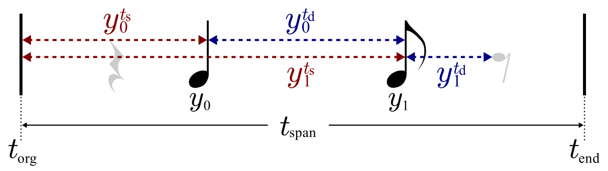

Figure 1. An example note sequence. Two note events y_0 and y_1 occur in the time-span between time origin t_{org} and end time t_{end}. The time-span duration t_{span} is three quarter-note periods. The rests at the start and end of the bar are not explicitly represented as objects in their own right here but as periods where no notes sound.

2.2 Note and velocity sequences

A single, *note* event y occurring in a time-span may be described by the tuple (t_s, t_d, ν) as shown in Figure 1, where t_s represents start or *onset* time relative to t_{org}, t_d represents note duration in the same units and ν represents the note *velocity* (i.e. the dynamic; how loud or accented the event is relative to others), where $\nu > 0$.

This allows us to represent an arbitrary rhythm as a *note sequence* Y, ordered in time

$$Y = \langle y_0, y_1, \cdots, y_{|Y|-1} \rangle \tag{1}$$

If TPQ is set to 4, an example note sequence representing the clave rhythm in Figure 2 could be:

$$Y = \langle (0,3,\mathbf{2}), (3,1,1), (6,2,\mathbf{2}), (10,2,1), (12,4,1) \rangle, \tag{2}$$

the higher velocity values of the first and third note tuples (in bold) showing that these are accented notes in this example.

An alternative representation of a rhythm is the *velocity sequence*. This is a sequence of values representing equally spaced points in a time-span; each value corresponding to the normalised velocity of a note onset if one is present or zero otherwise. The velocity sequence for the note sequence in Equation 2 can therefore be represented as

$$V = \langle 1,0,0,0.5,0,0,1,0,0,0,0.5,0,0.5,0,0,0 \rangle. \tag{3}$$

It should be noted that the conversion between note sequence and velocity sequence is not commutative, because the note duration information is lost in the conversion. As a result, converting from velocity sequence to note sequence, an assumption must be made that note durations equal to the inter-onset-intervals. Converting the velocity sequence in Equation 3 back to a note sequence would therefore yield

$$Y' = \langle (0,3,2), (3,\mathbf{3},1), (6,4,2), (10,2,1), (12,\mathbf{4},1) \rangle, \tag{4}$$

which has different durations (in bold) for the second and fourth notes compared to the original sequence in Equation 2.

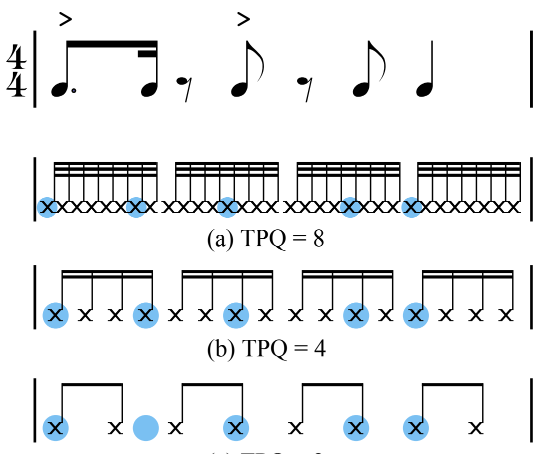

Figure 2. Representation of the *Son* clave rhythm at different Ticks Per Quarter-note (TPQ) resolutions. In (a) and (b) there is a tick for each note of the rhythm pattern thus all the sounded notes are captured (highlighted by the blue circles). However, in (c) where TPQ is 2, the second note of the pattern cannot be represented; the minimum resolution in this case is 4 TPQ.

2.3 Metrical structure and time-signature

Isochronous-meter is formed with a multi-level hierarchical metrical structure [20, 21]. The metrical hierarchy may be described with a *subdivision sequence* $\langle \lambda_0, \lambda_1, ..., \lambda_{L_{max}} \rangle$ such that in each metrical level L, the value λ_L specifies how nodes in the level above (i.e. $L-1$) should be split to produce the current level (see Figure 3). Any time-signature can be described by specifying a subdivision sequence and the metrical level that represents the beat.

Events at different metrical positions vary in perceptual salience or *metrical weight* [22]. These weights may be represented as a *weight sequence* $W = \langle w_0, w_1, ... w_{L_{max}} \rangle$. The prevailing hypothesis for the assignment of weights in the metrical hierarchy is that a time point that exists in both the current metrical level and the level above is said to have a *strong* weight compared to time points that are not also present in the level above [20]. The hierarchy of weights and subdivisions forms a key component in the prediction value calculation for many syncopation models. The choice of values for the weights in W can vary between different models but the assignment of weights to nodes at a given level in the hierarchy, as described in [20], is common to all.

2.4 Syncopation models

In this section we briefly review each implemented syncopation model, discussing their general hypothesis and giving a flavour of their mechanism. It is not possible to go into the full details of each implementation here but a thorough review of the models is given in chapter 3 of [15]. To help compare the capabilities of different models, we also give an overview of the musical features each one captures in Table 1.

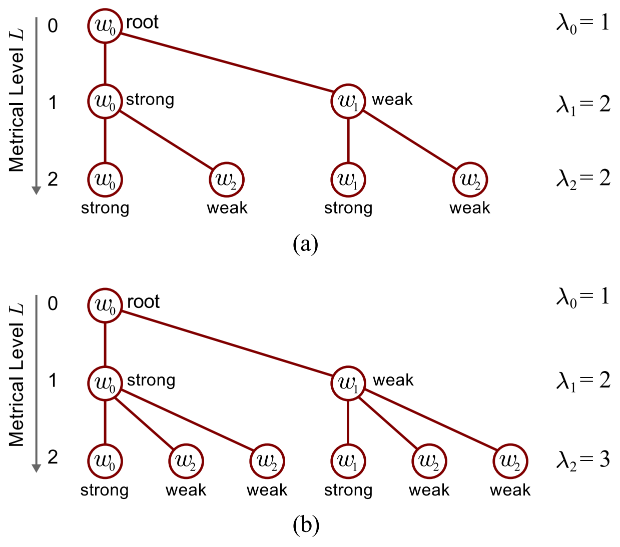

Figure 3. Metrical hierarchies for bars two time-signatures: (a) A simple-duple hierarchy dividing the bar into two groups of two (as with a 4/4 time-signature); (b) A compound-duple hierarchy dividing a bar into two beats, each of which is subdivided by three (e.g. 6/8 time-signature).

2.4.1 *Longuet-Higgins and Lee 1984 (LHL)*

Longuet-Higgins and Lee's model [1] decomposes rhythm patterns into a tree structure as described in Section 2.3 assigning metrical weights $w_L = -L$ i.e. $W = \langle 0, -1, -2, ... \rangle$. The hypothesis of this model is that a syncopation occurs when a rest (R) in one metrical position follows a note (N) in a weaker position. Where such a note-rest pair occurs, the difference in their metrical weights is taken as a local syncopation score. Summing the local scores produces the syncopation prediction for the whole rhythm sequence.

2.4.2 *Pressing 1997 (PRS)*

Pressing's cognitive complexity model [2, 16] specifies six prototype velocity sequences and ranks them in terms of *cognitive cost*. For example, the lowest cost is the *null* prototype for rhythms that contain either a single rest or note; a higher cost is given to the *filled* prototype that has a note in every position of the sequence e.g. $\langle 1, 1, 1, 1 \rangle$. The highest cost is given to the *syncopated* prototype that has a rest in the first (i.e. strongest) metrical position e.g. $\langle 0, 1, 1, 1 \rangle$. The model analyses the cost for the whole rhythm-pattern and for each of its sub-sequences at every metrical level determined by the subdivision factor. The final output is a sum of the costs per level weighted by the number of sub-sequences in each.

2.4.3 *Toussaint 2002 'Metric Complexity' (TMC)*

Toussaint's metric complexity measure [3] defines the metrical weights as $w_L = L_{\max} - L + 1$, thus stronger metrical positions are associated with higher weights and the weakest position will be $w_{L_{\max}} = 1$. The hypothesis of the model is that the level of syncopation is the difference between the metrical simplicity of the given rhythm (i.e. the sum of the metrical weights for each note) and the maximum possible metrical simplicity for a rhythm containing the same number of notes.

Property	LHL	PRS	TMC	SG	KTH	TOB	WNBD
Onset	✓	✓	✓	✓	✓	✓	✓
Duration					✓		✓
Dynamics				✓			
Mono	✓	✓	✓	✓	✓	✓	✓
Poly					✓	✓	✓
Duple	✓	✓	✓	✓	✓	✓	✓
Triple	✓	✓	✓	✓		✓	✓

Table 1. Musical properties captured by the different syncopation models. All models use note onsets, but only two use note duration rather than inter-onset intervals. Only SG takes dynamics (i.e. variation in note velocity) into account. All models handle monorhythms but the four models based on hierarchical decomposition of rhythm patterns are unable to handle polyrhythmic patterns. All models can process both duple and triple meters with the exception of KTH that can only process duple.

2.4.4 *Sioros and Guedes 2011 (SG)*

Sioros and Guedes [4, 17] also use metrical hierarchy to determine syncopation. The main hypotheses are that humans try to minimise the syncopation of a particular note relative to its neighbours in each level of the metrical hierarchy, and that syncopations at the beat level are more salient than those that occur in higher or lower metrical levels.

The metrical weights for this model are $w_L = L$ i.e. $W = \langle 0, 1, 2, ... \rangle$. The syncopation for a note is a function of its velocity, its position in the hierarchy and the weights of the previous and next notes in the rhythm sequence.

2.4.5 *Keith 1991 (KTH)*

Keith's model [5] defines two types of syncopated events: a *hesitation*, where a note ends off the beat (assigned a value of 1) and *anticipation*, where a note begins off the beat (assigned a value of 2). Where a note exhibits both a hesitation and an anticipation, a *syncopation* is said to occur and the respective values are summed to give a total of 3. The start and end time are considered off-beat if they are not divisible by the nearest power of two less than the note duration.

2.4.6 *Toussaint 2005 'Off-Beatness' (TOB)*

The off-beatness measure [6] is a geometric model that treats the time-span of a rhythm sequence as a T-unit cycle. The hypothesis, as applied to syncopation, is that syncopated events are those that occur in 'off-beat' positions in the cycle; the definition of *off-beatness* in this case being any position that does not fall on a regular subdivision of the cycle length T, thus the model is unable to measure cycles where T is 1 or prime.

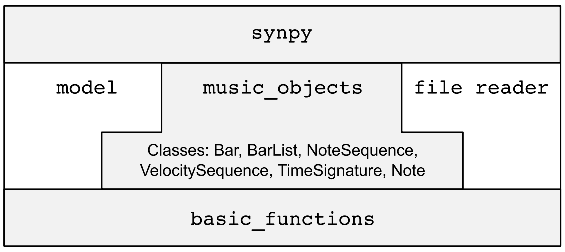
synpy
model
music_objects
file reader
Classes: Bar, BarList, NoteSequence, VelocitySequence, TimeSignature, Note
basic_functions

Figure 4. Module hierarchy in the SynPy toolkit: the top-level module provides a simple interface for the user to test different syncopation models. Musical constructs such as bars, velocity and note sequences, notes, and time-signatures are defined in the 'music objects' module; support for common procedures such as sequence concatenation and subdivision is provided in 'basic functions'. Models and file reading components can be chosen as required by the user.

2.4.7 Gómez 2005 'Weighted Note-to-Beat Distance' (WNBD)

The WNBD model of Gómez et al. [7] defines note events that start in between beats in the notated meter to be 'off-beat' thus leading to syncopation. The syncopation value for a note is inversely related to its distance from the nearest beat and is assigned more weight if the note crosses over the following beat.

3. FRAMEWORK

The architecture of the toolkit is shown in Figure 4. Syncopation values can be calculated for each bar in a given source of rhythm data along with selected statistics over all bars; the user specifies which model to use and supplies any special parameters that are required. Sources of rhythm data can be a bar object or a list of bars (detailed below in Section 3.1) or, alternatively, the name of a file containing music data. Where a model is unable to calculate a value for a given rhythm pattern, a 'None' value is recorded for that bar and the indices of unmeasured bars reported in the output. If no user parameters are supplied, the default parameters specified in the literature for each model are used. Output can optionally be saved directly to XML or JSON files. An example of usage in the Python interpreter is shown in Figure 5.

3.1 Music objects

The 'music objects' module provides classes to represent the musical constructs described in Section 2. A `Bar` object holds the rhythm information for a single bar of music along with its associated time-signature and optional tempo and TPQ values (see Section 2.1). `Bar` objects may be initialised with either a note sequence or velocity sequence and can be chained together in the form of a doubly-linked `BarList` allowing syncopation models to access next and previous bars where appropriate (several models [1, 2, 5, 7] require knowledge of the contents of previous and/or next bars in order to calculate the syncopation of the current bar). The note sequence and velocity sequence classes are direct implementations of the sequences described in Section 2.2. Common low-level procedures such as sequence concatenation and subdivision are provided in 'basic functions'.

```
>>>from synpy import *
>>>import synpy.PRS as model
>>>calculate_syncopation(model, "clave.rhy",
   outfile="clave.xml")
{'bars_with_valid_output': [0, 1],
 'mean_syncopation_per_bar': 8.625,
 'model_name': 'PRS',
 'number_of_bars': 2,
 'number_of_bars_not_measured': 0,
 'source': 'clave.rhy',
 'summed_syncopation': 17.25,
 'syncopation_by_bar': [8.625, 8.625]}
```

Figure 5. To use the toolkit, the top level `synpy` module is imported along with a model (in this example Pressing [2]). Calling `calculate_syncopation()` then gives the syncopation results as shown, also writing output to an XML file. Output file names and extra parameters for a model are added as optional arguments as required by the user.

```
T{4/4} # time-signature
TPQ{4} # ticks per quarternote
# Bar 1
Y{(0,3,2),(3,1,1),(6,2,2),(10,2,1),(12,4,1)}
# Bar 2
V{1,0,0,0.5,0,0,1,0,0,0,0.5,0,0.5,0,0,0}
```

Figure 6. Example rhythm annotation file `clave.rhy` containing two bars of the Son Clave rhythm as discussed Section 2. The first bar is expressed as a note sequence with resolution of four ticks per quarter-note; the second is the same rhythm expressed as a velocity sequence.

3.2 File Input

Two file reader modules are currently provided: one for for reading plain text rhythm annotation (`.rhy`) files and one for reading standard MIDI files (`.mid`). These modules open their respective file types and return a `BarList` object ready for processing.

Our `.rhy` annotation format is a light text syntax for describing rhythm patterns directly in terms of note and velocity sequences (see Figure 6). The full syntax specification is given in Backus Naur Form on the toolkit repository [19].

The MIDI file reader can open type 0 and type 1 standard MIDI files and select a given track to read rhythm from. Notes with zero delta time between them (i.e. chords) are treated as the same event for the purposes of creating note sequences from the MIDI stream. Time-signature and tempo events encoded in the MIDI stream are assumed to correctly describe those parameters of the recorded music so it is recommended that the user avoids incorrectly anno-

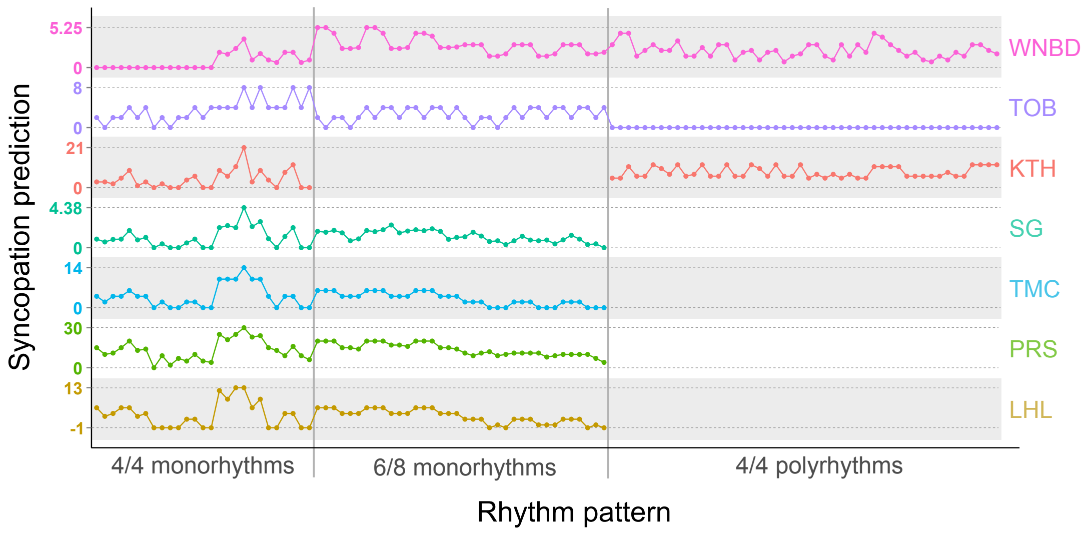

Figure 7. Syncopation predictions of the seven models in the toolkit for the syncopation dataset from [15]. For each model, the absolute range of prediction values is shown across all rhythm patterns in the dataset; ranges differing between models due to their different mechanisms. Within each rhythm category, the rhythm patterns are arranged by tatum-rate (i.e. quarter-note rate then eighth-note rate) then in alphabetical order (the data set naming convention uses letters a-l to represent short rhythm components that make up longer patterns). Gaps in model output occur where a particular model is unable to process the specific rhythm category i.e. LHL, PRS, TMC, SG cannot process polyrhythms and KTH can only measure rhythms in duple meters.

tated or unquantised MIDI files.

3.3 Plugin architecture

The system architecture has been designed to allow new models to be added easily. Models have a common interface, exposing a single function that will return the syncopation value for a bar of music. Optional parameters may be supplied as a Python dictionary if the user wishes to specify settings different from the those given in the literature for a specific model.

4. SYNCOPATION DATASET

The major outcome of the SynPy toolkit is to provide prediction of the level of syncopation of any rhythm pattern that can be measured by a given model. As a demonstration, we apply all seven syncopation models on the rhythms patterns used as stimuli for the syncopation perceptual dataset from [15, 23]. This dataset includes 27 monorhythms in 4/4 meter, 36 monorhythms in 6/8 and 48 polyrhythms in 4/4; altogether forming a set of 111 rhythm patterns.

Figure 7 plots the syncopation predictions of individual models for each rhythm. It presents the different ranges of prediction values for each model and shows their capabilities in terms of rhythm categories (refer to Table 1).

5. CONCLUSION

In this paper we have described SynPy, an open-source Python toolkit for calculating syncopation prediction values. We have introduced the theoretical concepts underpinning the toolkit and briefly reviewed the hypothesis and mechanism of the seven implemented models. The architecture of the toolkit has been introduced in Section 3 and an example of command line usage shown demonstrating ease of use. We have presented the syncopation predictions calculated by SynPy for the dataset from [15], providing an overall visualisation of the prediction ranges and capabilities of each individual model.

The SynPy toolkit possesses a number of merits, including the ability to process arbitrary rhythm patterns, convenient input from different sources of music data including standard MIDI files and text annotations, and output to XML and JSON files for further data analysis. It will be a valuable tool for many researchers in the computational music analysis community. It will be particularly useful to those who study syncopation models because it enables a level of comparison and testing for new models that was hitherto unavailable. The plugin architecture of the toolkit allows new models to be added easily in the future and open-source hosting in a repository on the soundsoftware.ac.uk servers ensures long term sustainability of the project.

Acknowledgments

This work was funded by the UK Engineering and Physical Sciences Research Council as part of the Soundsoftware Project based in the Centre for Digital Music at Queen Mary, University of London.

6. REFERENCES

[1] H. C. Longuet-Higgins and C. S. Lee, "The rhythmic interpretation of monophonic music," *Music Perception*, vol. 1, no. 4, pp. 424–441, 1984.

[2] J. Pressing, "Cognitive complexity and the structure of musical patterns," in *Proceedings of the 4th Conference of the Australian Cognitive Science Society*, 1997.

[3] G. T. Toussaint, "A mathematical analysis of african, brazilian, and cuban clave rhythms," in *Proceedings of BRIDGES: Mathematical Connections in Art, Music and Science*, 2002, pp. 157–168.

[4] G. Sioros and C. Guedes, "Complexity driven recombination of midi loops," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, 2011, pp. 381–386.

[5] M. Keith, *From Polychords to Pólya: Adventures in Music Combinatiorics*. Vinculum Press, 1991.

[6] G. T. Toussaint, "Mathematical features for recognizing preference in sub-saharan african traditional rhythm timelines," in *3rd International Conference on Advances in Pattern Recognition*, 2005, pp. 18–27.

[7] F. Gómez, A. Melvin, D. Rappaport, and G. T. Toussaint, "Mathematical measures of syncopation," in *BRIDGES: Mathematical Connections in Art, Music and Science*, 2005, pp. 73–84.

[8] P. E. Keller and E. Schubert, "Cognitive and affective judgements of syncopated musical themes," *Advances in Cognitive Psychology*, vol. 7, pp. 142–156, 2011.

[9] D.-J. Povel and P. Essens, "Perception of temporal patterns," *Music Perception*, vol. 2, no. 4, pp. 411–440, 1985.

[10] W. T. Fitch and A. J. Rosenfeld, "Perception and production of syncopated rhythms," *Music Perception*, vol. 25, no. 1, pp. 43–58, 2007.

[11] G. Madison, G. Sioros, M. Davis, M. Miron, D. Cocharro, and F. Gouyon, "Adding syncopation to simple melodies increases the perception of groove," in *Proceedings of: Conference of Society for Music Perception and Cognition*, 2013.

[12] M. A. G. Witek, E. F. Clarke, M. Wallentin, M. L. Kringelbach, and P. Vuust, "Syncopation, body-movement and pleasure in groove music," *PloS ONE*, vol. 9, no. 4, p. e94446, 2014.

[13] I. Winkler, G. P. Háden, O. Ladinig, I. Sziller, and H. Honing, "Newborn infants detect the beat in music," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, no. 7, pp. 2468–2471, 2009.

[14] P. Vuust, M. Wallentin, L. Ostergaard, and A. Roepstorff, "Tapping polyrhythms in music activates language areas," *Neuroscience Letters*, vol. 494, pp. 211–216, 2011.

[15] C. Song, "Syncopation: Unifying music theory and preception," Ph.D. dissertation, School of Electronic Engineering and Computer Science, Queen Mary, University of London, 2015.

[16] J. Pressing and P. Lawrence, "Transcribe: a comprehensive autotranscription program." in *Proceedings of the 1993 International Computer Music Conference*, 1993, pp. 343–345.

[17] G. Sioros, A. Holzapfel, and C. Guedes, "On measuring syncopation to drive an interactive music system," in *Proceedings of the 13th International Society for Music Information Retrieval Conference*, 2012, pp. 283–288.

[18] G. Sioros, "Kinetic. gestural controller-driven, adaptive, and dynamic music composition systems," http://smc.inescporto.pt/kinetic/?page_id=9, 2011.

[19] C. Song, C. Harte, and M. Pearce, "Synpy toolkit and syncopation perceptual dataset," https://code.soundsoftware.ac.uk/projects/syncopation-dataset, 2014.

[20] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. Cambridge, Mass: MIT Press, 1983.

[21] J. London, *Hearing in Time: Psychological Aspects of Musical Meter*. Oxford University Press, 2004.

[22] C. Palmer and C. L. Krumhansl, "Mental representations for musical meter," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 16, no. 4, pp. 728–741, 1990.

[23] C. Song, A. J. Simpson, C. A. Harte, M. T. Pearce, and M. B. Sandler, "Syncopation and the score," *PloS ONE*, vol. 8, no. 9, p. e74692. doi:10.1371/journal.pone.0074692, 2013.